STOCHASTIC DECISION PROCESSES IN LOCATION ANALYSIS

A THESIS

Presented to

The Faculty of the Division of Graduate

Studies and Research

by

Richard E. Rosenthal

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in the School of

Industrial and Systems Engineering

Georgia Institute of Technology

November, 1975

STOCHASTIC DECISION PROCESSES IN LOCATION ANALYSIS

Approved:

Donovan Young, Chairman

John A. White, Chairman

Paul S. Jones

V. E. Unger

Robert B. Cooper

Date Approved by Chairman: 11/13/75

ACKNOWLEDGMENTS

I would like to thank the faculty of the School of Industrial and Systems Engineering for their instruction and guidance. Donovan Young's contributions to this work and to my education in general are immense. Observing his knowledge and decision processes has affected my ways of thinking about problems. John White, whose suggestions established the direction of the dissertation, has been a model of professionalism. His efforts and concerns are deeply appreciated. Committee members Robert Cooper, Paul Jones, and V. Ed Unger have helped in this effort and throughout my graduate career. Other faculty with important contributions to my program were Professors Baker, Bazaraa, Jarvis, Rardin and Shetty.

I am grateful to fellow graduate students Luis Eduardo Contreras and Bruce Schmeiser for their interest in helping me organize and clarify my thoughts as this work progressed.

Thanks to Mandell Bellmore, Jon Liebman and Eliezer Naddor, exemplary teachers and scholars at Johns Hopkins who influenced my career choice.

The best aspect of coming to Georgia Tech was meeting Diana my first week in Atlanta. Thank you, Atlanta.

Knoxville, Tennessee
November, 1975

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

ABSTRACT

This research introduces methods of stochastic decision processes into location analysis. The specific model concerns making dynamic relocation decisions for a new facility (server) that must interact with existing facilities (customers) whose relocations are stochastic processes. The model is an infinite-horizon Markov decision chain whose solution gives a server relocation policy that minimizes the expected discounted sum of costs. Costs are location-dependent and are incurred in two ways: when the server makes choice relocations and when the server interacts with customers.

The model captures the essence of a variety of familiar dynamic location decision situations. Solutions generally have the server loosely pursuing real and anticipated customer moves, balancing the cost of relocating against the advantage of being favorably situated for future interaction costs. From a modeling standpoint, the model's significant limitations are an inability to accommodate interactions among customers (a customer's relocation probabilities depend on its own location only) and the fact that the associated exact solution procedures are not efficient for large numbers of customers and locations.

To reduce computational load and to allow solution of larger problems than could be solved using standard policy-iteration techniques, several methodological developments are presented: (1) a delayed-evaluation variant of policy iteration, (2) a partial feedback method that provides favorable timing of value updating in policy improvement,

(3) a natural-decomposition algorithm for rapid identification of ergodic structure of a Markov chain, (4) an efficient starting procedure based on a hypermyopic policy and (5) refinements in the numerical iterative methods of policy evaluation that exploit the special structure of the location problem. The first three of these developments apply not only to location problems but to Markov decision problems in general.

Computational experience is presented for problems with a single customer and up to 50 locations, and for problems with 3 customers and up to 6 locations. For single-customer problems, solution times grow more slowly than N^2, where N is the number of locations; thus large problems can be solved efficiently. For problems with M customers, solution times grow more rapidly than N^M; thus exact solution of large M-customer problems is not practical. A structured policy for M-customer problems, obtained by solving M separate single-customer problems, is presented. The structured policy is shown to give near-optimal results.

CHAPTER I

INTRODUCTION

1. Motivation

Location analysis, helping people decide where to put things, has established itself as an important branch of operations research. It has a history of interest dating from Fermat and a voluminous literature that no one person could safely claim to have absorbed entirely. It is strongly interdisciplinary: architects, city planners, civil and industrial engineers, environmentalists, health care specialists, military strategists, marketing analysts and political scientists are among those who have discovered a common need for rational methods of making facility-location decisions.

The common concept among various location analysis procedures, as demonstrated in the recent textbook by Francis and White [15] and in the important survey article by ReVelle, Marks and Liebman [39], is the formulation and solution (exact or approximate) of mathematical programming problems. Researchers in facility location have often applied existing mathematical programming techniques and have, additionally, developed new techniques that have found useful application outside of the facility-location context. It is well understood by leading developers of location analysis techniques that optimization with respect to a mathematical model is at best an aid to a decision maker's judgment and not a replacement for it. [39]

There are several reasons why the optimal solution to a mathematical program cannot be viewed necessarily as the optimal solution to the facility location situation modeled by the program. One difficulty, particularly in public-sector applications, lies in specifying an objective function that properly incorporates all of the important criteria by which a proposed solution ought to be judged. Other difficulties concern temporal aspects: how does one quantify future effects of present decisions, how does one treat uncertainty about future user characteristics for a facility whose location must be decided now, and how should facility locations be changed over time as user characteristics change?

The purpose of this thesis is to suggest approaches for resolving difficulties of this kind. The thrust is the application of *stochastic decision processes* to location analysis. Stochastic decision processes are decision-making methods that deal with systems (*processes*) whose behavior over time is dependent on the combined effects of chance (*stochastic*) and of choice (*decision*). We consider the dynamic location and relocation of a facility, called a *new facility* or a *server*, that must interact with each of a group of *existing facilities* or *customers* that themselves are being relocated dynamically. The new facility's relocation is choice-determined, while the existing facilities' relocation is considered to be chance-determined.

The key idea of stochastic decision processes is *sequential decision making*, in which decisions over time depend on new information as it is acquired. As explained by Veinott [52]:

> The new information acquired at a given point in time is the collection of values of the random variables which are first observed at that point and which are relevant to the choice of subsequent decisions. . . . One should make sequential decisions . . . [because] nothing is to be gained by committing oneself prematurely to a course of action.

The more common mathematical programming approaches that have been applied to dynamic facility location problems yield decision policies lacking a formal procedure for incorporating information not known at the time of solution. The research reported in this thesis is a first step in the development of stochastic decision processes for facility location, and the detailed results are limited to those cases in which the customer relocations are modeled by a stationary Markov chain. Thus this work could be more narrowly termed *Markov decision chains in location analysis*. It is hoped that this work will attract the interest of others, so that in the future the approach of stochastic decision processes will be applied to more complex and general facility location problems than those treated here.

2. Illustrative Examples

Three examples are presented here that illustrate the decision problem treated by this thesis: to determine where to locate and relocate a server that must interact with each of a group of existing facilities or customers whose own dynamic relocation is governed by a stationary Markov chain. The objective function is formulated so as to minimize the expected value of the present worth of all costs to be incurred over an infinite decision horizon. The operative tradeoff is between the cost of relocating the server and the savings of serving

from a more advantageous position. In typical decision problems of this kind, the costs of serving become too high if the server remains fixed, but on the other hand the costs of relocating become too high if the server tightly pursues the minimal-interaction-cost locations, so that an intermediate policy may be optimal. A typical solution has the server being relocated later than, less often than, and to a lesser extent than, the customers. If the customers' movements can be likened to those of the players in a basketball game, then the optimal movements of the server can be likened to those of the referee.

The model developed in this thesis is quite general and seems capable of capturing the essence of a wide range of familiar dynamic location decision situations that have not previously received formal treatment. At the large-scale end, one might consider such situations as the worldwide movements of fishing fleets, tactical movements of large military units or the movements of single-industry suppliers when the industry center shifts probabilistically with time, as the oil industry has shifted over the years from Los Angeles to Houston to Europe to the near East. At the small-scale end, one might consider the placement of a cursor or pointer so as to reduce data access times in a computer operating system. The three specific detailed examples to follow are much more ordinary in scale.

2.1 Example: Scheduled Maintenance

Consider a cycle of planned maintenance shutdowns of remotely situated plants in an oil refinery. The major work force will work on plants #1, #2 and #3, in order, proceeding to the next plant when work

on the previous plant is complete. The refinery owns a semi-mobile maintenance trailer to provide materials, supplies and temporary office space. Having the trailer at the same site as the major work force gives a baseline cost of 0 person-hr per day; if the trailer is at the home office (plant #0) the inconvenience is valued at 200 person-hr per day. The cost is 100 person-hr per day if the trailer is in the field while the major work force is in the field at a different plant. The time spent by the major work force at plant #1 is geometrically distributed with a mean of 2 days; at plant #2, geometrically distributed with a mean of 5 days; at plant #3, 1 day (days are not split between plants). At the end of each day, based on the current locations of the trailer and the work force, it must be decided where to locate the trailer. It costs the equivalent of 300 person-hr to move the trailer from any location to any other. The trailer begins and must end at plant #0.

This is an example of the kind of problem solved in Chapter III. Here the trailer is the server and the work crew is a single customer. The data provided by the preceeding description are the customer transition probability matrix, P , the server relocation cost matrix, F , and the server-customer interaction cost matrix, F′ , where

$$P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & .5 & .5 & 0 \\ 0 & 0 & .8 & .2 \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad F = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 300 & 300 & 300 \\ 300 & 0 & 300 & 300 \\ 300 & 300 & 0 & 300 \\ 300 & 300 & 300 & 0 \end{bmatrix} \end{array} \qquad F' = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 200 & 200 & 200 \\ \infty & 0 & 100 & 100 \\ \infty & 100 & 0 & 100 \\ \infty & 100 & 100 & 0 \end{bmatrix} \end{array}$$

The customer's starting location is plant #1. Plant #0 is modeled as an absorbing location because only one maintenance cycle is considered (in actual refineries each cycle is unique, and successive cycles are normally separated by intervals of 9 to 18 months). To insure that the server returns to plant #0 when the cycle is finished, the service cost is set at infinity when the server is in the field and the customer is not.

Note that this example not only provides an illustration of the kind of situation that can be handled, but also illustrates two important modeling techniques that demonstrate the great power of Markov-chain modeling: the use of a Markov chain to represent *scheduled*, as well as random, movements, and the use of a terminating Markov chain to represent what is essentially a finite-horizon situation without violating the infinite-horizon assumptions critical to efficient solution.

The state of this system on any day is an ordered pair (i,j) where i is the server location and j is the customer location. The procedure developed in Chapter III was applied to this problem, and the output was a matrix K where k(i,j) is the plant at which the server should be located on any given day that begins with the system in any given state (i,j) . For this example, the optimal solution is

$$K = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 2 & 2 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 3 & 0 \end{bmatrix} \end{array}$$

This solution, or *policy*, is optimal in the sense that it minimizes the total expected cost of the maintenance costs associated with location of the trailer. Since the starting state is (0,1), the first action is to move the trailer to plant #2, since element (0,1) of K is 2; the trailer should remain in plant #2 until the work crew reaches plant #3, and then should be moved back to the home office. This is a rather typical solution for problems dealt with in this thesis, in that the server loosely follows the (anticipated) movements of the customer(s).

If there were more than one work crew, each following its own Markov transition law and each having its own matrix of costs of interacting with the trailer, this problem would become one of the kind treated in Chapter IV. In that case, the optimal policy would be specified in the form $k(i,j_1,j_2,\ldots)$, giving the location to which the trailer should be moved when it and the work crews are observed to be in locations $i,j_1,j_2,\ldots$, respectively.

2.2 Example: Choosing Strategic Warehousing Locations

Consider a grocery-chain warehouse (server) located in Atlanta, (location #1), serving a large local distribution terminal (customer #1) in Atlanta, another (#2) in Savannah (location #2), and a group of individual supermarkets in various locations. Data on prospective profits and costs of doing business have been collected so that management can decide whether and when to relocate the warehouse to Savannah or possibly to Macon (location #3). The decision problem has become pertinent because it is believed that the two large customers may possibly relocate, customer #1 to Savannah, customer #2 to Macon. The potential

benefit of following or anticipating these moves must be weighed against the cost of relocation and the potential losses due to being farther away from the center of demand of the other customers.

Assume an annual decision review, a discount factor of $\beta=.85$, an annual relocation probability of .2 for customer #1 and .4 for customer #2. These probabilities are independent, and it is assumed that a relocated customer will not relocate again. The expected contributions to net annual profits for the various situations are as follows:

	Profit on Sales to Customer #1 if at:		Profit on Sales to Customer #2 if at:		
Server Location	Location #1	#2	Location #2	#3	Profit from Sales to Other Customers
1	63	41	39	38	22
2	42	65	75	55	14
3	40	40	43	68	10

The cost of relocating in Savannah is 150; in Macon, 200.

The Markov transition matrices for the two large customers are respectively,

$$P_1 = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{pmatrix} .8 & .2 \\ 0 & 1 \end{pmatrix} \end{array}, \qquad P_2 = \begin{array}{c} \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cc} 2 & 3 \end{array} \\ \begin{pmatrix} .6 & .4 \\ 0 & 1 \end{pmatrix} \end{array}$$

The expected contributions to net annual profit from sales to the small customers and the costs of moving the server are combined to define the

server relocation cost matrix

$$F = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} -12 & 136 & 190 \\ & -14 & 190 \\ & 136 & -10 \end{pmatrix} \end{array} ,$$

where negative costs indicate profits. The server-customer interaction cost matrices are F_1' and F_2' where

$$-F_1' = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{pmatrix} 63 & 41 \\ 42 & 65 \\ 40 & 40 \end{pmatrix} \end{array} , \qquad -F_2' = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cc} 2 & 3 \end{array} \\ \begin{pmatrix} 39 & 38 \\ 75 & 55 \\ 43 & 68 \end{pmatrix} \end{array} .$$

This is a multifacility problem of the kind treated in Chapter IV, having the optimal policy that, regardless of the movement of customer #2, the warehouse should follow customer #1 to Savannah.

The complete output of the algorithm of Chapter IV for this problem is as follows:

Current Location: Warehouse	Customer #1	Customer #2	Optimal Warehouse Move	Net Present Worth	Remarks
1	1	2	Stay at 1	735.97	Do not anticipate
1	1	3	Stay at 1	730.73	Do not follow #2
1	2	2	Move to 2	765.41	Follow #1
1	2	3	Move to 2	740.92	Follow #1
2	1	2	(Stay at 2)	857.91	(Unreachable)
2	1	3	(Stay at 2)	833.42	(Unreachable)
2	2	2	Stay at 2	915.41	After relocation
2	2	3	Stay at 2	890.92	After relocation
3	1	2	(Stay at 3)	756.03	(Unreachable)
3	1	3	(Stay at 3)	784.54	(Unreachable)
3	2	2	(Move to 2)	765.41	(Unreachable)
3	2	3	(Stay at 3)	784.54	(Unreachable)

As is true for dynamic programs in general, the algorithm solves the problem for all possible states, not all of which can necessarily be reached by the optimal policy from a given starting state. This example problem provides a clear illustration of the *extensive structure* of the dynamic, probabilistic location problems treated in this thesis. Fully half of the states cannot be reached from the given starting state in this example; thus the values of the unreachable states are immaterial in calculating the values of the other states, and consequently the computational load of the algorithm is substantially reduced. Typically, as in this problem, some states are excluded by zeroes in the transition matrices, and others are excluded by each decision policy.

2.3 Example: Strategic Picker Relocations in an Automated Warehouse

Consider a 3-bin automated warehouse with bins located in a line. A programmed picker moves clockwise only down the line from bin #1 to bin #3 and behind the bins back to bin #1:

To service a demand the picker moves from its current location to the demand location, picks up the demanded item and carries it to the dump point, which is at bin #1. Optionally, the picker may then be relocated so as to be in an advantageous position for servicing the next demand. The costs of service moves are as shown by the small numbers in the diagram, and the costs of relocations are 1/10 of these numbers. Thus, for example, the picker may be relocated to bin #3 at a cost of .2; if the next demand is at bin #2, the service cost is 6, since the picker must proceed from bin #3 clockwise 3 units to pick up the item and 3 more units to dump it at bin #1.

Demands are highly autocorrelated; the probability that the next demand is at bin ℓ, given that the previous demand was at bin j, is

j \ ℓ	1	2	3
1	.5	.4	.1
2	.2	.7	.1
3	.4	.3	.3

To minimize expected costs per demand in the long run, what is the optimal relocation policy, and what is the resulting expected cost?

This is a single-facility problem of the kind treated in Chapter III, with the picker as the server and the demand as the customer. At a discount factor of $\beta=.9999$, the methods of Chapter III yield an optimal policy of relocating the picker to bin #2 whenever the previous demand was at bin #2, and leaving the picker at the dump point (bin #1) otherwise. The discounted expected cost of this policy is between 27137 and 27140, depending on the initial demand, so the long-run expected cost per demand is approximately $(1-\beta)(27138) \simeq 2.714$.

A corresponding multifacility problem amenable to the methods of Chapter IV could be formulated by having more than one type of demanded item at each bin location, with different demand autocorrelations for each type.

3. Relation to the Location Analysis Literature

The work in this thesis lies within the area of location analysis not only because the problems treated are formulated with the express intent of modeling situations that arise when locations need to be chosen, but also because the resulting formulations themselves lead naturally back to the location context. In context-free nomenclature, the mathematical entities we call new facilities could be called *choice entities*, existing facilities *chance entities* and location *condition*. Relocation costs of choice entities could be called *condition-change costs*, and *interaction costs* would simply become any costs that depend on the combination of the conditions of all entities. However, since

location is a very common attribute that determines interaction costs between entities, a location interpretation (or at least a location analogy) arises almost inevitably for such problems.

To see where the present work fits into the location-analysis framework, we can consider the problem characteristics in relation to the taxonomy proposed by Francis and White [15]. We treat single (as opposed to multiple) new facilities, with point (as opposed to area) locations, and the locations of the new facilities are decision variables (as opposed to parameters); the interactions between new and existing facilities are quantitative, location dependent, dynamic and probabilistic (as opposed to qualitative, location independent, static and deterministic), and the interactions are described by parameters (as opposed to decision variables); the existing facility locations are dynamic, probabilistic, point quantities treated as parameters (as opposed to static, deterministic, area quantities treated as decision variables); the solution space is multidimensional and discrete, the distance measures can be rectilinear, Euclidian or "other," and the objective is to minimize total cost. The work thus fits neatly into the location-analysis taxonomy, at least in the sense that the taxonomy neither omits important attributes nor asks for attributes not possessed.

Not all niches in the Francis and White taxonomy are equally populated with published work, and the niche described above is particularly vacant. In comparison with published work, the two distinguishing features of the problems considered here are 1) probabilistic location of existing facilities and 2) dynamic relocation of both new and existing facilities. Both of these features have been incorporated,

although separately, in the relevant published work to be described in the next two sections.

3.1 Stochastic Location Literature

Probabilistic considerations have been included in the analysis of static location problems reported by several authors. These authors have frequently used as their starting points certain well-studied deterministic models, such as the single-facility minimal-sum location problem [1,9,26,27], a multifacility minimal-sum location problem [1,48], a plant location problem [24,25] and an emergency service facility location problem [5].

The deterministic formulation of the single-facility minimal-sum location problem is

Given:
1. a set of existing facilities with known locations P_i, $i=1,\ldots,m$, where P_i is either a point in R^2 or a node of a network;
2. a solution space X of possible locations for the new facility;
3. a measure of distance d where $d(X,P_i)$ for $X \epsilon X$ is the distance between possible new-facility location X and existing-facility location P_i; and
4. a weight or interaction cost w_i, $i=1,\ldots,m$, representing a cost incurred for each unit of distance separating the new facility and existing facility i;

find $X \epsilon X$ so as to

$$\text{minimize} \quad f(X) = \sum_{i=1}^{m} w_i d(X,P_i) \quad .$$

This problem has been treated extensively under a variety of assumptions on d and X. The solution methods for these problems constitute the fourth chapter of a location-analysis textbook by Francis and White [15].

The first probabilistic version of the single-facility location problem is credited to Cooper [9], who relaxed the condition that the locations P_i be known constants. He assumed instead that each P_i, $i=1,\ldots,m$, was a random variable having a bivariate normal probability distribution. He developed a globally convergent iterative method to minimize the expected value of $f(X)$ when d is the Euclidian metric and $X=R^2$. One of Cooper's main findings was that the optimal coordinates of the new-facility location for the probabilistic problem were different from the optimal solution of the corresponding deterministic problem where the P_i were replaced by their means. Katz and Cooper [27] solved the problem of minimizing $E[f(X)]$ when d is Euclidian, $X=R^2$ and the locations have exponential or double-exponential distributions. Aly [1] solved the problem for the cases of rectilinear and squared Euclidian distances and introduced chance constraints on the distance separating the new facility from the existing facility.

Probabilistic versions of minimal-sum multifacility problems have been solved by Aly [1] and Seppala [48]; the former author assumed that existing-facility locations but not interaction costs were random variables; the latter, the reverse. Plant location problems with uncertain customer demands have been treated by Hurter and Prawda [24] and by Jucker and Carlson [25]. The present work does not involve

multiple new facilities.

By introducing chance constraints, Chapman and White [5] extended the set-covering formulation of the emergency facility location problem. In this problem the number of new facilities (ambulances) that will adequately serve the existing facilities (accident victims) is the objective function to be minimized. Chapman and White [5] also introduced and solved a minimax probabilistic emergency facility location problem. The objective was to minimize the maximal probability of a victim's not being covered by an ambulance. In both types of problem, randomness was assumed in the ability of an emergency vehicle to travel quickly enough to succeed in serving.

3.2 Dynamic Location Literature

Dynamic or multiperiod facility location problems treated in the literature seem always to assume deterministic problem data. Roodman and Schwarz [40] give optimal and heuristic solutions to a multiperiod version of the Effroymson and Ray plant location problem [13]. The locations and demands of all customers in all periods must be known in advance in order to use their procedure. Wesolowsky and Truscott [55] give the solution to the zero-one version of the same problem. Other researchers in this area are Ballou [2], Scott [44], Sheppard [47], Tapiero [50] and Wesolowsky [54].

It is to be hoped that the results of this thesis will make some start towards the establishment of a branch of the location analysis literature that will treat problems that are both probabilistic and dynamic.

4. Summary

Chapter II of this thesis introduces, outside the location analysis context, a special form of stochastic decision processes called here *state-change Markov decision chains*. A new derivation of the equations for determining the expected present worth of the costs in a valued Markov chain is presented. Then we define a decision process based on directive intervention with a randomly evolving process. The relationship of this new process to the standard Markov decision chain is shown, justifying the application of known results to the new process. Chapter II also contains a survey of some of the relevant literature concerning stochastic decision processes.

In Chapter III the state-change Markov decision chain is used to provide a natural model for single-facility dynamic probabilistic location problems such as the scheduled maintenance problem given in this chapter. By exploiting special characteristics of the problem we develop an efficient exact procedure. Extensions of the model that this algorithm can easily handle are also presented. In addition a new algorithm called *natural decomposition* for evaluating the states of a non-ergodic Markov chain is presented.

Chapter IV concerns a more difficult multifacility problem such as the warehouse location problem in this chapter. Extension of the exact single-facility approach is developed, and its computational performance reported. Approximate procedures are developed for large problems of this type. The solutions obtained from one such approximation, a computationally efficient structured policy, are compared to the optima. The final chapter proposes ideas for extending the work of this thesis.

CHAPTER II

STATE-CHANGE MARKOV DECISION CHAINS

This chapter is devoted to state-change Markov decision chains, a specialization of Markov decision chain methodology, that will be applied to dynamic probabilistic location problems in subsequent chapters.

The organization of the chapter is, broadly, 1) analysis of the valued Markov chain, the purely probabilistic process that a) underlies the state-change Markov decision chain and b) constitutes a useful descriptive tool, (Section 1); 2) explanation of the state-change Markov decision chain (Sections 2,3,4); and 3) a survey of some important developments in Markov decision chains with their application to the state-change case (Sections 5,6).

The valued Markov chain and the state-change Markov decision chain are directly traceable to previous researches. The former can be viewed as a direct consequence of Markov's own work [23,Appendix B]; the latter is a straightforward specialization of Markov decision processes.* There exists an extensive literature on Markov decision processes which clarifies the important theoretical issues such as existence of pure, stationary, β-optimal policies and the convergence properties of several

---

* The name popularized by Howard [22], *Markov decision process*, for the entity referred to here has begun to be replaced in the literature by the more appropriate name *Markov decision chain* [37,52]. *Markov decision process* is still used in reference to decision problems whose underlying probabilistic structure is a continuous Markov process. Not modified by "Markov decision" the word "process" is used in this thesis as a general undefined term, much like "system."

solution procedures. By establishing the relationship between the state-change formulation and the standard formulation of Markov decision chains (in Section 5), we are then able to apply the known results to the state-change case (in Section 6).

1. Valued Markov Chains

The fundamental probabilistic process underlying the state-change Markov decision chain is the *valued Markov chain*. A valued Markov chain consists of

1) a Markov chain X_t , $t=0,1,2,\ldots$, on a countable state space S , which makes transitions according to a law of motion $P=\{p_{ij}\}$, $i,j\epsilon S$; and

2) a cost (or reward) system $C=\{c_{ij}\}$, $i,j\epsilon S$, $p_{ij}>0$ such that a payment in the amount c_{ij} is expended (or received) whenever a transition from state i to state j occurs. c_{ij} is called the *state-transition cost* or, when $c_{ij}=c_i$ for all $j\epsilon S$, the *state-visit cost*.

The valued Markov chain describes an infinite horizon process which generates a stream of payments*

$$c(X_0,X_1),\ c(X_1,X_2),\ldots,c(X_{t-1},X_t),\ldots.$$

Valued Markov chains have a wide range of modeling applicability.

---

* Throughout the thesis, the elements of an array such as C will be represented by the corresponding miniscule letter with row and column identified either in "functional" form, $c(i,j)$, or subscripted c_{ij} ; the choice depending entirely on typographical considerations. The guideline is to use subscripts except when subscripted subscripts would be necessary. For example, we use c_{ij} but $c(X_{t-1},X_t)$.

In general the information one seeks to obtain is the "value" of the stream of payments. Interpretation of this "value" may vary from case to case. In order to avoid the uninteresting case where the stream of payments is valued at infinity, one of two approaches, the *terminating process* or the *discounted process*, is ordinarily used. The former approach is to define the states $i \in S$, the transition probabilities p_{ij}, and the transition costs c_{ij} so that the Markov chain has probability 1 of eventually hitting an absorbing state or ergodic class of states after which all payments generated are zero. The second approach for avoiding infinitely valued cash streams is to employ the common economic practice of discounting whereby a payment c_t at time t is valued at the *present worth* $c_t\beta^t$ where β, the *discount factor*, is a known constant between zero and one. The *present worth* (or *discounted value*) of the process is $\sum_{t=1}^{\infty} c_t\beta^t$. This value is both probabilistic and dependent on the initial state X_0 of the system. Thus, for each $i \in S$ an expression v_i is defined as the *expected present worth of the process given state* i *as the starting state.** To solve a valued Markov chain is to solve the system of linear equations, or *value equations*,

$$v_i = \sum_{j \in S} p_{ij}(c_{ij} + \beta v_j) \quad , i \in S \; . \tag{1.1}$$

Each equation states that v_i is equal to the sum of an expected

---

* In the terminology of the recent theory of stochastic processes, e.g. Çinlar [6], $(v_1, v_2, \ldots)$ is a *potential* of the Markov chain $X_1, X_2, \ldots$.

immediate payment (or *expected transition cost*), r_i , where

$$r_i = \sum_{j \in S} p_{ij} c_{ij} \quad , \tag{1.2}$$

and a discounted expected value for all future payments, $\sum_{j \in S} p_{ij} \beta v_j$. From (1.2), an equivalent form of (1.1) is

$$v_i = r_i + \beta \sum_{j \in S} p_{ij} v_j \quad , \; i \in S \; . \tag{1.3}$$

In the undiscounted, terminating case equations (1.1) and (1.3) apply with $\beta=1$.

1.1 Derivation of the Value Equations

Value equations (1.1) or (1.3) are justified in the literature by either 1) a statement that they are intuitively obvious or 2) a limit argument based on analagous equations for a finite horizon [22,p.82;21, p.552]. An alternative derivation follows.

By definition the expression v_i in (1.1) and (1.3) is the conditional expectation given $X_0=i$ of the random variable

$$v = c(X_0,X_1) + \beta\, c(X_1,X_2) + \beta^2\, c(X_2,X_3) + \ldots \; ; \tag{1.4}$$

that is, if it exists,

$$v_i = E[v | X_0 = i] \; . \tag{1.5}$$

Let μ_{it} be the conditional expectation given $X_0 = i$ of the cost incurred at the t^{th} transition, i.e.,

$$\mu_{it} = E[c(X_{t-1},X_t) \mid X_0 = i] \quad , i\epsilon S,\ t=1,2,.... \tag{1.6}$$

Then

$$v_i = \mu_{i1} + \beta\,\mu_{i2} + \beta^2\,\mu_{i3} + ... \tag{1.7}$$

The key step in the derivation of (1.1) is a recursive characterization, (1.8) and (1.13), of the μ_{it}'s .

$$\mu_{i1} = \sum_{j\epsilon S} p_{ij} c_{ij} \quad , i\epsilon S \ . \tag{1.8}$$

For, $i\epsilon S$, $t=1,2,...$

$$\mu_{i,t+1} = E[c(X_t,X_{t+1}) \mid X_0 = i\,]$$

$$= \sum_{k\epsilon S} \sum_{\ell\epsilon S} \Pr\{X_{t+1} = \ell,\ X_t = k \mid X_0 = i\} c_{k\ell}$$

$$= \sum_{k\epsilon S} \sum_{\ell\epsilon S} p_{ik}^{(t)}\, p_{k\ell}\, c_{k\ell} \quad , \tag{1.9}$$

where $p_{ik}^{(t)}$ is the t-step Markov transition probability from i to k . By defining

$$p_{ik}^{(0)} = \begin{cases} 1 & \text{if } i=k \\ 0 & \text{if } i\neq k \end{cases} \quad , i,k\epsilon S \tag{1.10}$$

equations (1.8) and (1.9) may be combined as

$$\mu_{i,t+1} = \sum_{k\epsilon S} p_{ik}^{(t)} \sum_{\ell\epsilon S} p_{k\ell}\, c_{k\ell} \quad , t=0,1,2,.... \tag{1.11}$$

From (1.10) and the Chapman-Kolmogorov equations

$$p_{ik}^{(t)} = \sum_{j\in S} p_{ij} p_{jk}^{(t-1)} \quad , \; i,k\in S, \; t=1,2,.... \tag{1.12}$$

Inserting (1.12) in (1.11) yields

$$\mu_{i,t+1} = \sum_{j\in S} p_{ij} \sum_{k\in S} p_{jk}^{(t-1)} \sum_{\ell\in S} p_{k\ell}\, c_{k\ell} \quad , \; i\in S, \; t=1,2,...,$$

from which follows the recursion

$$\mu_{i,t+1} = \sum_{j\in S} p_{ij}\mu_{jt} \quad , \; i\in S, \; t=1,2,.... \tag{1.13}$$

To complete the derivation,

$$v_i = \mu_{i1} + \sum_{t=1}^{\infty} \beta^t \mu_{i,t+1} \qquad \text{[by (1.7)]}$$

$$= \sum_{j\in S} p_{ij} c_{ij} + \sum_{t=1}^{\infty} \beta^t \sum_{j\in S} p_{ij}\mu_{jt} \qquad \text{[by (1.8) and (1.13)]}$$

$$= \sum_{j\in S} p_{ij} c_{ij} + \beta \sum_{j\in S} p_{ij} \Big(\mu_{j1} + \sum_{t=1}^{\infty} \beta^t \mu_{j,t+1}\Big)$$

$$= \sum_{j\in S} p_{ij} c_{ij} + \beta \sum_{j\in S} p_{ij} v_j \qquad \text{[by (1.7)]}$$

$$= \sum_{j\in S} p_{ij} \big(c_{ij} + \beta v_j\big) \quad , \; i\in S \; .$$

The valued Markov chain is useful for solving many operations research problems involving discrete probability. Some of these problems involve streams of payments directly whereas other problems lend

themselves more subtly to advantageous use of cost modeling. Examples of valued Markov chains are given below.

1.2 Examples

Machine replacement. A group of machines, each one of which is either operative or inoperative at any time, is inspected every N years. When operative a machine generates an income of $\$\bar{A}$ per year at a uniform continuous rate. There is a probability 1-p of its failing during the year. If inoperative a machine generates no income and is instantly replaced at the time of inspection at a cost of \$R. The expected present worth of an initially operative machine will be determined at a yearly discount rate β or nominal continuous-compound interest rate $r = -\ell n\beta$.

The transition matrix for the state of a machine at the time just prior to inspection is

$$P = \begin{matrix} \\ \text{op} \\ \text{inop} \end{matrix} \begin{pmatrix} & \\ p^N & 1-p^N \\ p^N & 1-p^N \end{pmatrix} ,$$

the second row reflecting the effect of replacement. If a machine survives the entire inspection interval its income is $\frac{\bar{A}}{r}(1-e^{-rN})$, discounted to the beginning of the interval. If a machine does not survive the inspection interval, then it will earn $\$\bar{A}$ per year continuously until some time t and zero afterwards, where t is a truncated exponentially distributed random variable on the interval (0,N) [60]. The expected present worth (with respect to the beginning of the interval) of this cash flow with uncertain timing [62] is $\frac{\bar{A}}{r}\left(1-\frac{1-e^{-(r+1/m)N}}{(rm+1)(1-e^{N/m})}\right)$

where m is the expected time to failure when the inspection interval increases without bound. Thus

$$C = \begin{Bmatrix} c_1 & c_2 \\ c_1-R & c_2-R \end{Bmatrix}$$

where

$$c_1 = \frac{\overline{A}}{r}(1-e^{-rN})$$

$$c_2 = \frac{\overline{A}}{r}\left[1 - \frac{1-e^{-(r+1/m)N}}{(rm+1)(1-e^{N/m})}\right] .$$

It is clear that $v_{op} = v_{inop}+R$ because, after the replacement cost is charged, all future cash flows are the same for both starting states. Therefore, only one value equation is necessary. Setting $v = v_{op}$ and again invoking (1.1),

$$v = p^N(c_1+\beta v) + (1-p^N)\,[c_2 + \beta(v+R)]$$

or

$$v = \frac{p^N c_1 + (1-p^N)(c_2+\beta R)}{1-\beta} .$$

Additional modeling power can be achieved for this type of problem with the valued Markov chain by including one or more intermediate states between operative and inoperative. [12,p.122] This would be useful when the machine is known to have a nonconstant failure rate.

Inventory depletion. Inventory of a particular item is taken at the beginning of each month. There is no replenishment and stock is depleted according to a random demand per month, y, taking on values $y=0,1,2,\ldots$ with probability distribution $p_0,p_1,p_2,\ldots$. When outgoing inventory j for the month is positive a holding cost h_j is charged. The shortage cost is $Q(y-i)$ when y is greater than i, the beginning-of-month inventory. The expected long-run cost at monthly discount rate β is required, given starting inventory N.

Let $r_j = \sum_{y=j}^{\infty} p_j$, $u_j = \sum_{y=j+1}^{\infty} Q(y-a)p_j$. For $N=4$, say, the transition probability matrix for beginning-of-month inventory is

$$P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ r_1 & p_0 & 0 & 0 & 0 \\ r_2 & p_1 & p_0 & 0 & 0 \\ r_3 & p_2 & p_1 & p_0 & 0 \\ r_4 & p_3 & p_2 & p_1 & p_0 \end{pmatrix} \end{array} \tag{1.14}$$

and the transition cost matrix

$$C = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} u_0 & & & & \\ h_1+u_1 & h_1 & & & \\ h_2+u_2 & h_2 & h_2 & & \\ h_3+u_3 & h_3 & h_3 & h_3 & \\ h_4+u_4 & h_4 & h_4 & h_4 & h_4 \end{pmatrix} \end{array} \tag{1.15}$$

The resulting value equations are equivalent to

$$v_0(1-\beta) = u_0$$

$$v_1(1-\beta p_0) = h_1 + r_1 u_1 + \beta r_1 v_0$$

$$v_2(1-\beta p_0) = h_2 + r_2 u_2 + \beta r_2 v_0 + \beta p_1 v_1$$

$$\vdots$$

$$v_N(1-\beta p_0) = h_N + r_N u_N + \beta r_N v_0 + \beta \sum_{i=1}^{N-1} p_i v_{N-i}$$

which may be solved sequentially by substitution.

This example will be reconsidered in Section 2.3 when replenishment (state-change) is allowed.

2. Introducing Decision Variables: State-Change Markov Decision Chains

The valued Markov chain has been used for descriptive purposes, such as to evaluate the expectation of a random variable or the probability of an event. Prescriptive use, in decision problems, is confined to situations where alternative policies for operating a system can be enumerated, with each policy defining its own valued Markov chain. Through exhaustive calculation of the value for each chain, the most desirable policy can be determined.

A more systematic approach allowing for a wide range of applications is the *state-change Markov decision chain*. In this model a decision maker intervenes with the random state transitions by imposing a

state change whenever he chooses. In all practical cases it is advantageous to change the system by choice no more than once between successive chance transitions, because changing the system incurs a cost and all other cash flows take place after a chance transition. The state-change Markov decision chain is, then, a discrete stochastic process whose transitions are alternatively governed by a decision rule and a Markov transition matrix.

As in the valued Markov chain, let X_t be the state of the system at the time of the t^{th} chance transition, let P be the law of motion describing chance transitions when they occur, and let C be the system of costs generated by chance transitions. If the decision maker observes $X_t=i$ then his state-change decision may be specified by the rule:

For all $i,k\epsilon S$, $t=1,2,\ldots$

$d_{ik}^t = 1$ if the decision maker chooses to change the state of the system to state k immediately following the t^{th} chance transition, given that $X_t=i$,

$= 0$ otherwise.

An alternative, randomized decision rule is given by

d_{ik}^t = Prob [decision maker chooses to change the state of the system to state k immediately following the t^{th} chance transition $|X_t=i$]

where the conditions

$$\sum_{k\epsilon S} d_{ik}^t = 1 \quad , \text{ all } i,t$$

$$0 \leq d_{ik}^{t} \quad , \text{ all } i,k,t$$

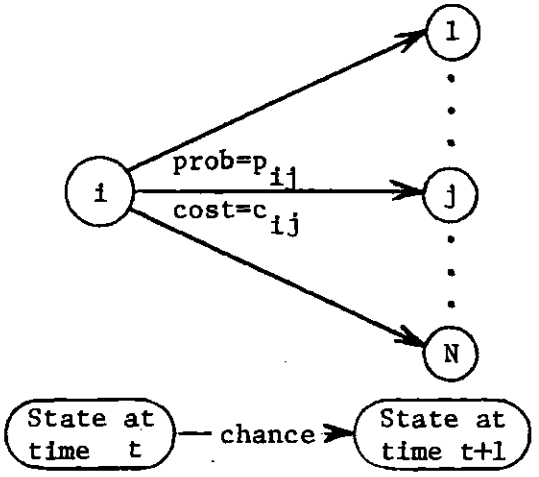
are enforced.

2.1 Stationary Policies and the Complete Transition Matrix

As will be shown, there is really very little difference between the two interpretations of d_{ik}^{t}, as binary-valued expressions or as probabilities. Further, as is justified for all discounted, infinite-horizon Markov decision processes, consideration is limited to stationary decision rules. Stationary decision rules have the property that for all $i,k \epsilon S$, $d_{ik}^{1}=d_{ik}^{2}=\ldots=d_{ik}^{t}=\ldots$. In such cases where state changes are independent of time the superscript is supressed.

The cumulative effect of chance and choice transitions under a stationary policy can be summarized by the *complete transition matrix* T, $T=DP$, where D is the matrix of elements d_{ik}. That is, the process $X_t;t=0,1,2,\ldots$, is a Markov chain and the conditional probability $\text{Prob}[X_{t+1}=j|X_t=i] = t_{ij} = \sum_{k \epsilon S} d_{ik}p_{kj}$. This relationship is depicted graphically in Figure 1(b).

While the state-change Markov decision chain is a Markov chain, its analysis or even its modeling utility is different from that of a valued Markov chain because the complete transition matrix contains decision variables d_{ik} whose values, a priori, are not known. In fact, analysis of the process is concerned primarily with determining the best choices for those decision variables. The meaning of "best choices" is explained in Section 3. First the definition of state values analagous to equation (1.1) for valued Markov chains is considered.



$$(\text{transition probability})_{ij} = p_{ij}$$

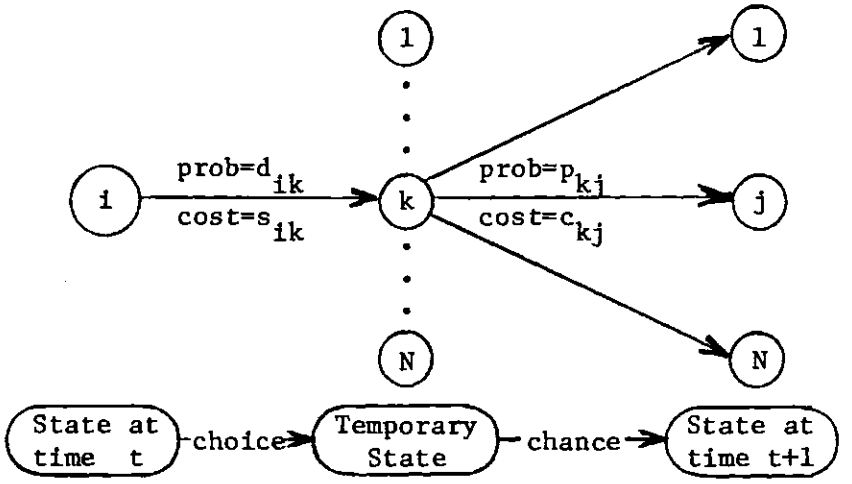
$$(\text{expected transition cost})_i =$$

$$r_i = \sum_{j \in S}^{N} p_{ij} c_{ij}$$

Figure 1(a).

Valued Markov Chain:

Purely Probabilistic



$$(\text{transition probability})_{ij} = \sum_{k \in S} d_{ik} p_{kj}$$

$$(\text{expected transition cost})_i = r_i' = \sum_{k \in S} d_{ik} (s_{ik} + \sum_{j \in S} p_{kj} c_{kj})$$

$$= \sum_{k \in S} d_{ik} (s_{ik} + r_k)$$

Figure 1(b).

Introducing Decision Variables (d_{ik}) :

the State-Change Markov Decision Chain

2.2 Value Determination

Once the decisions, D, are specified the state-change process is a Markov chain with transition matrix T, as noted. Moreover, the process is a valued Markov chain.

Let s_{ik} equal the *state-change cost*, the immediate cost incurred whenever the decision is made to change the state from i to k. The expected transition costs r_i' for the valued Markov chain associated with operating the state-change process under specified decisions D are

$$r_i' = \sum_{k \in S} d_{ik}(s_{ik} + \sum_{j \in S} p_{kj} c_{kj}) \quad . \tag{2.1}$$

This cost is the sum of the expectations of the state-change cost and the succeeding chance transition cost. (See Figure 1(b).)

The state values for the state-change Markov decision chain given decisions D are obtained by substituting t_{ij} and r_i' in (1.3) for p_{ij} and r_i, respectively:

$$v_i = r_i' + \beta \sum_{j \in S} t_{ij} v_j$$

$$= \sum_{k \in S} d_{ik}(s_{ik} + \sum_{j \in S} p_{kj} c_{kj}) + \beta \sum_{j \in S} \sum_{k \in S} d_{ik} p_{kj} v_j$$

or $$v_i = \sum_{k \in S} d_{ik}[s_{ik} + \sum_{j \in S} p_{kj}(c_{kj} + \beta v_j)] \quad , \; i \in S \; . \tag{2.2}$$

If a pure policy is used (i.e., if all d_{ik} are 0 or 1) and if k_i denotes the state to which the process is moved whenever the decision-maker observes it in state i, then equations (2.2) are

equivalent to

$$v_i = s(i,k_i) + \sum_{j\in S} p(k_i,j)[c(k_i,j) + \beta v_j] \quad , \; i\in S \; . \qquad (2.3)$$

This equation simply says that the state value v_i is equal to the cost of changing the state to k_i plus the expected present worth of all subsequent cash flows.*

Solution of equations (2.2) is the *value determination operation* for the state-change Markov decision chain. When it is necessary to indicate explicitly that values obtained through this operation are with respect to particular decisions D, the notation $v_i(D)$ will be employed. The notation $K = \{k_i\}$ will be used to denote a pure, stationary state-change policy.

2.3 Example

When replenishment is allowed in the inventory system of section 1.2 the system becomes a state-change Markov decision chain. A decision-maker can intervene with the chance-dependent (i.e., demand-dependent) transitions by procuring new stock and, hence, changing the state. Let $M(x)$ be the cost of procuring x units for $x>0$. Then the state-change costs are

---

* The expected present worth of subsequent cash flows is not $v(k_i)$. The reason for this is that the state value v_i determined by (2.2) is the expected present worth of the process starting in state i and starting when it is the decision-maker's "turn" to change the state. The expected present worth starting in state i when it is Nature's "turn" to change the state is a different quantity. For state k_i it is precisely the quantity $\sum_{j\in S} p(k_i,j)[c(k_i,j) + \beta v_j]$.

$$s_{ik} = \begin{cases} M(k-i) & , \quad k>i \\ 0 & , \quad k=i \\ \infty & , \quad k<i \end{cases} \qquad i,k=0,1,2,.... \tag{2.4}$$

Let $S = \{0,1,...,N\}$ where N is the capacity of the system, i.e. $h_i=\infty$ for $i>N$. Then the state values associated with any stationary replenishment policy D are characterized by (2.2), (1.14), (1.15) and (2.4).*

3. Choice of an Objective Function: β-Optimality

This section deals with the question of defining "optimal decisions" for the state-change Markov decision chain. The set of feasible stationary policies is the set $\mathcal{D}$ of all stochastic matrices having the same order as S.

If there is a clearly defined starting point to the process and one has knowledge of the initial state distribution,

$$\pi_i(0) = \Pr\{X_0=i\} \qquad , \; i\in S \; ,$$

then an appropriate choice for the objective function with respect to which an optimal policy $D\in\mathcal{D}$ should be chosen is

$$\text{Minimize} \quad f_1(D) = \sum_{i\in S} \pi_i(0)\, v_i(D) \; .$$

---

* This model is accurate for any form of the demand distribution and the holding, shortage and procurement functions. More direct methods are applicable if these functions have special form. [16,Ch.4]

If one has no information about initial conditions, then, an appropriate choice for the objective function might be

$$\text{Minimize} \quad f_2(D) = \sum_{i \in S} \pi_i(D)\, v_i(D)$$

where $\pi_i(D)$, $i \in S$ is the stationary distribution of the state-change Markov decision chain operating under policy D , if it exists uniquely. $f_2(D)$ is difficult to work with due to the dependence of $\pi_i(D)$ on $T=PD$ which is the transition matrix of a chain which need not be ergodic.

Remarkably, in order to optimize either $f_1(D)$ or $f_2(D)$ it is not necessary to know $\pi_i(0)$ or $\pi_i(D)$, because f_1 , f_2 and all other nonnegative linear combinations of the v_i 's have the same minimizing policy. In fact, there exists a policy D^* which simultaneously minimizes each v_i . By definition a policy D^* is *β-optimal*, optimal for a given discount factor β , if $v_i(D^*) = \min_{D \in \mathcal{D}} v_i(D)$ for each $i \in S$.†

Existence of β-optimal policies for all discounted Markov decision chains is proven in [41].* It is worth pointing out that existence of β-optimal policies is an extraordinary phenomenon; because, generally, one cannot simultaneously optimize several functions of the same decision

---

† If necessary, the minimum may be replaced by the infimum, but this is not needed when, as assumed here, all costs s_{ik} , c_{ij} are bounded.

* That state-change Markov decision chains are, in fact, Markov decision chains is shown in Section 5 of this chapter.

variables.* For this reason, the definition of "optimal" is a key issue in the study of multi-objective optimization. Although Markov decision chains are multi-objective optimization problems (minimize v_1 , minimize v_2 , ... , minimize v_N) the literature does not treat them in that context.** This is because the N minimization problems have a common optimal policy. In other words, β-optimality, when it exists, is an irrefutably sensible approach to multi-objective optimization.

---

* Here is an example in location analysis where two objective functions of the same decision variables have different optimizing solutions. Consider the multifacility location problem [15,Ch.4] where it is required to find the locations $\underline{X} = (X_1,\ldots,X_n)$ of n new facilities so as to minimize the sum of a cost

$$\phi_1(\underline{X}) = \sum_{j=1}^{n} \sum_{k=j+1}^{n} v_{jk}\, d(X_j,X_k) \quad ,$$

pertaining to new-facility interactions (where v_{jk} is the cost per unit distance for interaction between new facilities j and k , and $d(\cdot,\cdot)$ is the distance measure), and a cost

$$\phi_2(\underline{X}) = \sum_{j=1}^{n} \sum_{i=1}^{m} w_{ji}\, d(X_j,P_i) \quad ,$$

pertaining to interactions between the new facilities and m existing facilities located at points $P_1,\ldots,P_m$. In some applications it may not be appropriate to add the two different kinds of costs together into a single objective. For instance, if the same maintenance crew is to maintain all n new facilities, then rather than wishing to minimize $\phi_1 + \phi_2$ we might wish to minimize

$$\phi_3(\underline{X}) = \operatorname*{Maximum}_{j,k}\, d(X_j,X_k) \quad ,$$

the maximum distance between new facilities, and also (separately) minimize ϕ_2 . Clearly ϕ_3 is minimized when all new facilities are located coincidentally, which, in any practical case, is a suboptimal solution for ϕ_2 .

** A recent paper [53] on the "multiple criterion Markov decision process" introduced the situation where the costs incurred at each transition are vector-valued.

4. Extremal Equations and Policy Iteration

Not only are β-optimal policies existent and stationary they are also pure, that is, the elements of D^* are 0 or 1. Hence, as noted previously, there is very little difference between the two interpretations of decision variables d_{ik}, as binary variables or as probabilities. This can be shown in two ways: by showing that the linear programming formulation of the problem using continuous variables (the probabilities interpretation) has an all integer optimal solution [10], or by invoking Bellman's principle of optimality [21], from which the following *functional* or *extremal* equations are intuited.†

$$v_i(D^*) = \min_{k \in S} \left\{ s_{ik} + \sum_{j \in S} p_{kj}(c_{kj} + \beta v_j(D^*)) \right\} , \quad i \in S . \qquad (4.1)$$

The pure stationary β-optimal strategy satisfies $d_{ik^*} = 1$ for k^*, the minimizing k in (4.1). If more than one state-change decision provides the minimum, the choice of k^* is arbitrary; in such cases there are alternative (pure) β-optimal policies and all convex combinations of these constitute other (randomized) β-optimal policies.

Howard's celebrated *policy iteration algorithm* [22] is well suited for solving state-change Markov decision chains. It forms the basis of the analysis used to solve the first dynamic, probabilistic problem in location analysis. It is described here in the context of a state-change process with finite state space $S = \{1,2,\ldots,N\}$. The algorithm is:

---

† A proof of the extremal equations is given by Ross [41,p.121-2].

Step 0: Choose an initial pure policy D .

Step 1: (Value determination operation.) Solve the valued Markov chain associated with the current policy. I.e., solve for v_i ,

$$v_i = \sum_{k=1}^{N} d_{ik}[s_{ik} + \sum_{j=1}^{N} p_{kj}(c_{kj} + \beta v_j)] \quad , \; i=1,\ldots,N \; .$$

For notation purposes (and storage economy) let k_i be the current decision in state i , that is k_i is the state for which $d(i,k_i) = 1$ in the current policy. Then the system of equations is written

$$v_i = s(i,k_i) + \sum_{j=1}^{N} p(k_i,j)[c(k_i,j) + \beta v_j] \; , \; i=1,\ldots,N. \quad (4.2)$$

Step 2: (Policy improvement routine.) Using the values just computed for v_i , $i=1,\ldots,N$, revise the policy by setting k_i equal to the minimizing state in the expression

$$\min_{k=1,\ldots,n} [s_{ik} + \sum_{j=1}^{N} p_{kj}(c_{kj} + \beta v_j)] \quad (4.3)$$

for each $i=1,\ldots,N$. If Step 2 results in a change of policy go to Step 1, otherwise the current policy is optimal.

Convergence of the policy iteration algorithm to a β-optimal policy is assured for Markov decision chains. [41] We appeal to this well-known result in asserting the convergence of the above procedure for state-change chains. To make this assertion it is necessary, of course, that we show 1) that a state-change Markov decision chain is a Markov decision chain and 2) that the algorithm above is, in fact, Howard's algorithm for the state-change case. These facts are established

in the next section.

5. The Relationship Between the State-Change Formulation and the Standard Formulation of Markov Decision Chains

The purpose of this section is to show that the state-change process is a special case of a Markov decision chain. It will enable us to make use of established results from the literature of Markov decision processes.

Prior to this section the state-change process was viewed as a natural extension of valued Markov chains wherein choice-transitions are permitted. It was shown that once a stationary policy is specified the state-change process is itself a valued Markov chain. This led to the value determination equations (2.2) which follow from the value equations (1.3) derived from first principles in Section 1.1. It was not until we considered prescribing "optimal" policies that we began to rely on the theory of Markov decision chains. Specifically, we claimed the existence of pure stationary β-optimal policies, characterized them with extremal equations (4.1) and suggested finding them by policy iteration. The remainder of this section demonstrates the relationship which justifies these claims, namely, that the state-change process is the special case of a Markov decision chain in which the state space and the action space are identical.

The standard formulation of the discounted Markov decision chain is next presented. Except for some notational changes and other minor modification this presentation is taken directly from Veinott [52].

"Consider a system which is observed at each of a sequence of

points in time labeled $0,1,2,\ldots$. At each of those points the system is found to be in one of N *states*, labeled $1,2,\ldots,N$.... Intuitively, the state of the system at a particular point in time provides all information relevant for subsequent decisions. At each point in time at which the system is observed in state i, an action a is chosen from a finite set A_i of possible actions and a cost $r(i,a)$ [to be discounted by a known factor β to present worth at time 0]* is incurred. The conditional probability that the system is observed in state j at time $N+1$ given that it is found in state i at time N, that action a is taken at that time, and given the observed states and actions taken at times $1,2,\ldots,N-1$, is assumed to be a nonnegative function $p(j|i,a)$ depending only on j, i and a [with $\sum_{j=1}^{N} p(j|i,a) = 1$]†.

"If the action chosen in each state at each point in time depends only on the state and the point in time, the successive states of the system form a (finite) Markov chain. For this reason, we call the above system a (finite) *Markov decision chain.*

"A Markov decision chain can be represented as a directed graph with labeled arcs in which the nodes are partitioned into two sets, one consisting of the set of all states and the other the set of all actions available in at least some state. There is an arc from each state to each action available in that state, the arc label being the cost incurred by the given state-action combination. Moreover, there is an arc

---

* Veinott does not assume discounting. Rather he assumes there is a probability $q(i,a)$ that the proccess will enter a *stopped* state after which all costs are zero. The two formulations are equivalent if $q(i,a) = \beta$ for all i,a.

† In Veinott's presentation, $\sum_{j=1}^{N} p(j|i,a) = 1-q(i,a) \leq 1$.

from each action (in a state) to each state to which there is a positive transition probability, the arc label being the indicated probability in this case." [52]

An example from [52] with states displayed as circles and actions as triangles is given in Figure 2. There are two actions in state 1: the first costs 10^6 and leads with probability 1 to state 2, the second costs 0 and leads to states one and two each with probability 1/2 . State 2 has only one action; it costs 2 and leads to state 2 with probability 1 .

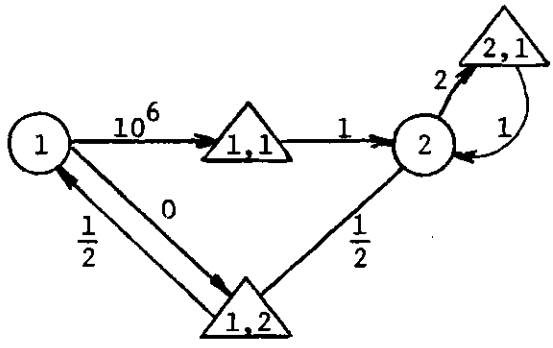


Figure 2.

Directed Graph Representation of a Markov Decision Chain

Veinott makes the important point that when an action leads to a given state with probability one, we can omit the node corresponding to that action, while maintaining a direct arc from the state where the action may be taken to the state where the action will lead. This arc is labeled with the associated cost. For instance the two actions (1,1) and (2,1) both lead to state 2 with probability one so the graph of Figure 2 should be replaced by the graph of Figure 3.

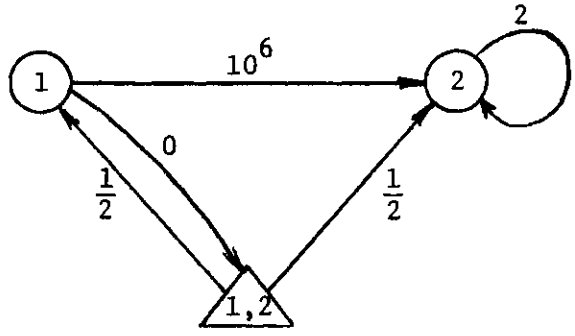


Figure 3.

Abbreviated Graph Representation of a Markov Decision Chain

This point about deleting action nodes is precisely the concept underlying the connection between the state-change and the standard formulations. We can think of action 1 in state 1 of the example as a deliberate or choice-determined state-change. If all actions have the property that they lead with probability 1 to a certain state then we can call the process a state-change Markov decision chain. When all the actions are of this type it becomes difficult to represent the system with Veinott's graph and we have to revert to a graph such as in Figure 1b, where states at time t are distinguished from states at time $t+1$.

Another way of saying that all actions are state-changes is to say that the state space is identical to the action space. With this in mind we put the state-change process into standard formulation by writing actions as integers $k=1,\ldots,N$, transition probabilities as

$$p(j|i,k) = p_{kj} \quad , \; i,j,k=1,\ldots,N \; ; \tag{5.1}$$

and (expected) costs

$$r(i,k) = s_{ik} + r_k \quad , \; i,k=1,\ldots,N \; . \tag{5.2}$$

In (5.1) and (5.2) the left-hand-terms come from the standard formulation notation, the terms on the right have the state-change context meaning given earlier in the chapter. (See Figure 1.)

Notice from (5.1) that for all $j,k=1,\ldots,N$, $p(j|i,k) = p(j|2,k) = \ldots = p(j|N,k)$. Thus the state-change formulation has a significantly smaller data gathering and storage requirement than the standard formulation.

To complete this section we note that the extremal equations for the standard formulation [41],

$$v_i = \min_a \left[r(i,a) + \beta \sum_{j=1}^{N} p(j|i,a)\, v_j \right] \qquad i=1,\ldots,N , \qquad (5.3)$$

are equivalent to (4.1) when specialized to the state-change case. This follows immediately from (5.1), (5.2) and

$$r_k = \sum_{j=1}^{N} p_{kj}\, c_{kj} \qquad , \; k=1,\ldots,N$$

which is simply (1.3).

Since the existence of pure stationary β-optimal policies and Howard's policy iteration algorithm [22] are based on (5.3), and since (4.1) is the state-change equivalent of (5.3), it is also easy to show that the algorithm in Section 4 is Howard's policy iteration algorithm for finding a β-optimal policy in the state-change case.

6. Solution Techniques

The remainder of this chapter is essentially the literature survey for the part of this thesis that deals with Markov decision chains.

Existing ideas and methods which have proven useful for solving dynamic, probabilistic problems in location analysis in the later chapters of this thesis are highlighted. In all cases results are presented in the context of the state-change formulation; of course, equations (5.1) and (5.2) are all that is necessary to transform the results back, as they originally appeared, to standard formulation.

The organization of the survey is as follows: Section 6.1 describes several iterative methods for finite-state, finite-action discounted Markov decision chains. The unified framework (equation (6.23)) in which these methods are presented is new. Section 6.2 focuses on the work of MacQueen [31,32] for identification of nonoptimal actions. Section 6.3 discusses other existing methods not covered by the framework of Section 6.1 and briefly mentions how the Markov decision chain model has been generalized. Section 6.4 discusses the important topic of *structured* policies that can be specifically designed for particular problems.

In 1973 Schweitzer [43] compiled a bibliography on Markov decision processes with more than 600 entries. The current survey adds several papers which have appeared since 1973.

6.1 Two Operators and Several Iterative Numerical Methods

By introducing some operator notation we can encapsulate several existing algorithms in a single equation. Both of these operators are automappings on the space of bounded real N-tuples, which in the present context are candidate values for solution of the extremal equations. The operators are used to revise these values whenever they are nonoptimal. An N-tuple will be written as $V = (v_1,\ldots,v_N)$.

The *policy improvement operator* Π maps V to ΠV where

$$(\Pi V)_i = \min_k \left\{ s_{ik} + r_k + \beta \sum_{j=1}^{N} p_{kj} v_j \right\} \quad , \; i=1,\ldots,N \; . \tag{6.1}$$

A way of interpreting ΠV is to think of its i^{th} component as the minimal expected present worth if the process starts in state i and is to terminate after one period with a final discounted cost of βv_j when the final state is j. The extremal equations (4.1) that characterize the optimal set of values V^* are equivalent to

$$(\Pi V^*)_i = v_i^* \quad , \; i=1,\ldots,N \; ,$$

or

$$\Pi V^* = V^* \; . \tag{6.2}$$

In other words V^* is a fixed point of the mapping Π. Denardo [11]† discovered that the mapping Π belongs to a certain class of functions (called contractions) that are guaranteed to have a unique fixed point, thus proving the existence of β-optimal policies.# These functions have the additional property that if they are applied successively to any starting solution the result will tend to the fixed point. That is, for any bounded $U = (u_1,\ldots,u_n)$

---

† The results reported in [11] pertain to a class of stochastic decision processes that subsumes discounted finite-state Markov decision chains.

Another proof of the existence of a solution to (6.2) is given by Shapiro [46] who demonstrates that Π is a continuous function on a compact convex set; thus, by the Brouwer theorem, Π possesses a fixed point.

$$\Pi^n U \to V^* \quad \text{as} \quad n \to \infty \quad , \tag{6.3}$$

where Π^n is the composition of Π with itself n times. An additional property of Π, Denardo [ ] discovered, is its monotonicity,†

$$U \geq V \implies \Pi U \geq \Pi V \quad . \tag{6.4}$$

The consequence of (6.4) is that the convergence of (6.3) is always monotone, i.e.,

$$U \geq \Pi U \implies \Pi U \geq \Pi(\Pi U) = \Pi^2 U \implies \Pi^n U \downarrow V^* \tag{6.5}$$

or

$$U \leq \Pi U \implies \Pi U \leq \Pi^2 U \implies \Pi^n U \uparrow V^* \tag{6.6}$$

Case (6.5) applies when the starting solution is greater than V^*. Case (6.6) applies when the starting solution is less than V^*.

The second operator introduced is the *policy evaluation operator* Λ. The operator Λ is defined with respect to a particular policy $D = \{d_{ik}\}$ or, in the case of a pure policy, $K = \{k_i\}$. In all cases the policy assumed is the last one proposed. Λ maps V to ΛV where

$$(\Lambda V)_i = \sum_{k=1}^{N} d_{ik}\left(s_{ik} + r_k + \beta \sum_{j=1}^{N} p_{kj}\, v_j\right) \quad , \; i=1,\ldots,N \tag{6.7}$$

or, in the case of a pure policy (which is always assumed at this point),

---

† An N-tuple inequality such as $U \geq V$ means $u_i \geq v_i$ for all $i=1,\ldots,N$.

$$(\Lambda V)_i = s(i,k_i) + r(k_i) + \beta \sum_{j=1}^{N} p(k_i,j)\, v_j \quad , \; i=1,\ldots,N \; . \tag{6.8}$$

The interpretation of ΛV is the same as that of ΠV except that the specified policy, not the minimizing policy, must be used in each step. The important convergence property of Λ [41,p.123] is that, for any U

$$\Lambda^n U \to V(D) \quad \text{as} \quad n \to \infty \; . \tag{6.9}$$

That is, the successive application of Λ is asymptotically equivalent to the value determination operation. When a value determination operation is performed by direct matrix inversion we shall use the notation Λ^∞ to indicate this action. That Λ shares the properties of Π, including monotonicity of the convergence in (6.9), is apparent when one realizes that Λ is equivalent to the special, "degenerate," case of Π when only one action is permissible at each state (i.e. all other actions (state-changes) have infinite cost.)

Howard's policy iteration algorithm, according to the operator notation, is

$$V^{(n+1)} = (\Lambda^\infty \Pi) V^{(n)} \quad , \tag{6.10}$$

where the superscripts on V's are iteration numbers. Equation (6.10) says that $V^{(n)}$ is first improved by a policy improvement and then the newly obtained policy is evaluated by a complete value determination operation. If we take $V^{(0)} = 0$, then $\Pi V^{(0)}$ will result in the *myopic policy*, the policy which minimizes single period costs, and $V^{(1)}$ is the N-tuple of state values associated with that policy.

A class of algorithms may be expressed as

$$v^{(n+1)} = (\Lambda^y \Pi^z) v^{(n)} \tag{6.11}$$

where y and z are integers, $y \geq 0$, $z \geq 1$. In addition to Howard's policy iteration algorithm, where $y=\infty$ and $z=1$, equation (6.11) represents the multi-step type algorithms of Denardo [11], Schweitzer [42] and Totten [51] where more than one policy improvement is made between successive policy evaluations. The case $y=0$, $z=1$,

$$v^{(n+1)} = \Pi v^{(n)} , \tag{6.12}$$

corresponds to White's [57] algorithm or successive approximation,* wherein value determination is completely ignored. The cases $0<y<\infty$, $z=1$ correspond to algorithms that incorporate approximate, iterative value determinations. Another important case, proposed by Porteus [35], uses $y=\infty$ and $z=z_n$ at iteration n, where z_n is the number of times Π must be applied to $v^{(n)}$ in order for the same policy to be specified twice in a row.

The implicitly assumed termination criterion for (6.11) is $v^{(n+1)} = v^{(n)}$, or, acknowledging the unavoidable errors inherent in digital computation, $\text{maximum}_i |v_i^{(n+1)} - v_i^{(n)}| < \xi$ for some positive real ξ. This condition of repeated (or very nearly repeated) values is called here *value convergence*. Value convergence for all cases of (6.11) constitutes

---

* Another name for (6.12) is value iteration [22,51] which is somewhat misleading for a method that only improves policies and never evaluates them.

the solution of the Markov decision process. This fact, i.e. that

$$(\Lambda^y \Pi^z)V = V \quad \Rightarrow \quad \Pi V = V \quad ,$$

follows from the montonicity property.† Howard's policy iteration algorithm and Hasting's slight modification of policy iteration (see (6.14)) terminate when Π produces the same policy twice in a row. This criterion, which we shall call *policy convergence*, is not generally sufficient for convergence to V^* when $y<\infty$ or $z>1$. An algorithm incorporating the suggestion of Porteus [ ] to apply Π as many times as required for policy convergence and incorporating the sufficiency of policy iteration in Howard's policy iteration method is flowcharted in Figure 4. We shall refer to procedures of this type as *delayed evaluation algorithms*, since their central feature is delaying policy evaluation until after policy convergence.

---

† Precisely, $V^{(1)} = \Pi V$, $V^{(2)} = \Pi V^{(1)}$, ... , $V^{(z)} = \Pi V^{(z-1)}$, $V^{(z+1)} = \Lambda V^{(z)}$, ... , $V^{(z+y)} = \Lambda V^{(z+y-1)} = V$ imply $V^{(1)} = V^{(2)} = \ldots = V^{(z+y)} = V$. $z \neq 0$ is assumed.

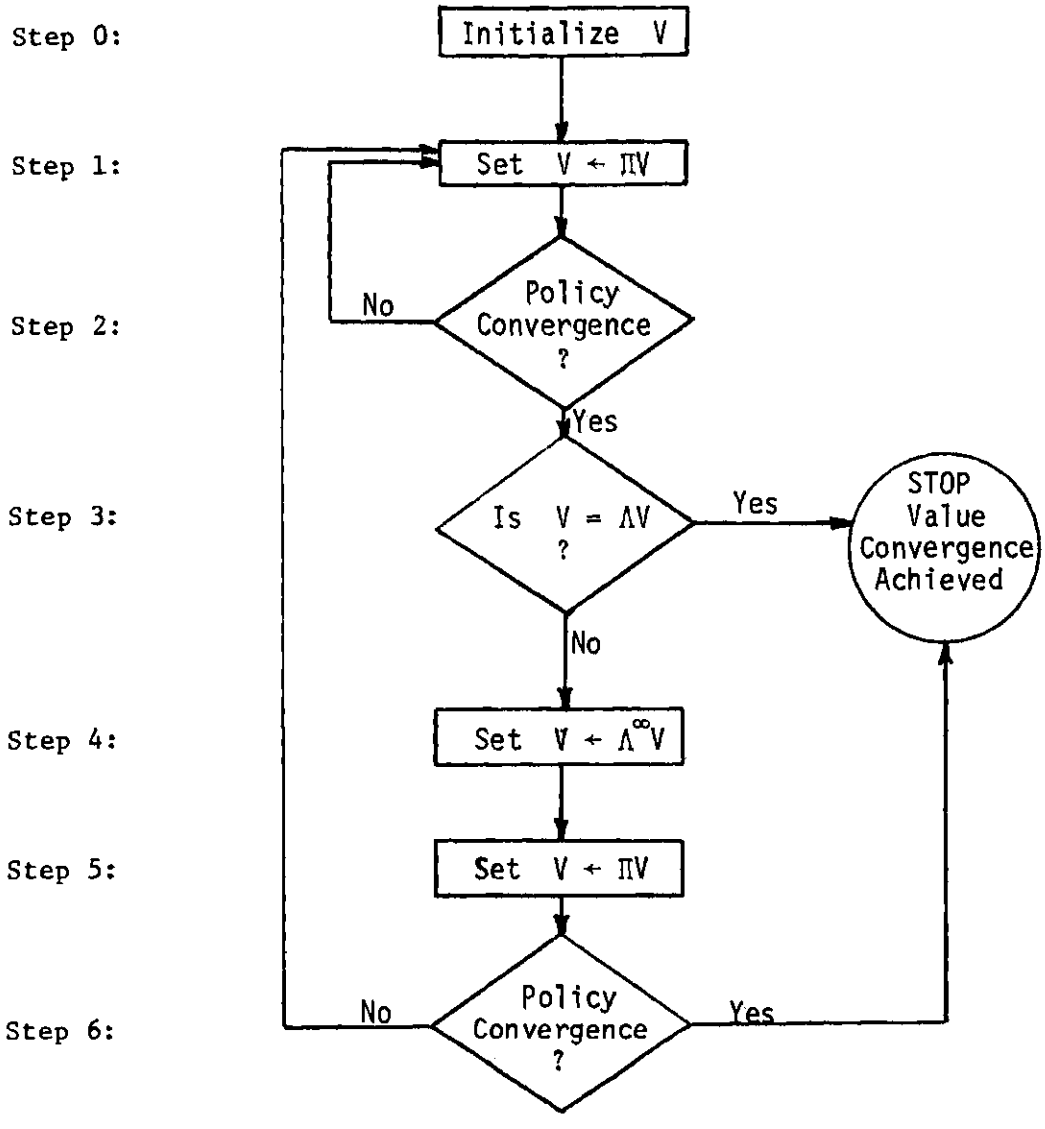


Figure 4.

Delayed Evaluation Algorithm

Termination at Step 3 is through verification of value convergence by direct calculation. Terminating with policy convergence at Step 6 is justified because Steps 4 and 5 constitute a single iteration of Howard's policy iteration method. Any of the methods (6.11) (or (6.23)) may be used in Step 1. The initialization of V in Step 0 may be done in

several ways such as:

$$v^{(0)} = \Pi(0) \tag{6.13}$$

$$v^{(0)} = (\Lambda^{\infty}\Pi)(0) \tag{6.14}$$

$$v^{(0)} = \Pi\left\{\frac{1}{1-\beta}\Pi(0)\right\} \tag{6.15}$$

where 0 is the N-tuple of all zeroes, and $\frac{1}{1-\beta}\Pi(0)$ is $\Pi(0)$ after each component is multiplied by $\frac{1}{1-\beta}$. In (6.13) $v^{(0)}$ is simply the vector of smallest immediate costs. In (6.14) $v^{(0)}$ is the vector of state values for the myopic policy generated by (6.13). The components of $v^{(0)}$ in (6.15) are the best expected present worths assuming the process operates probabilistically for one period and then, if it reaches state j, it remains in state j forever. Figure 4 represents an algorithmic structure, a synthesis of several existing concepts, that can take many particular forms; it is used in Chapter III.

Many of the other contributions to Markov decision processes consist of slight refinements to (6.11). Several proposed algorithms have the structure of (6.11) with modified versions of Π and Λ. These modifications arise from application of standard concepts of iterative numerical analysis. We can think of Π as a Jacobi [56] or simultaneous displacement procedure [7]. A Gauss-Seidel [56] or successive displacement [7] policy improvement operator is $\tilde{\Pi}$ where*

---

* Whenever the upper limit of a summation is less than the lower limit, the summation is taken to be zero. This avoids dealing with many special cases.

$$(\tilde{\Pi}V)_i = \min_k \left\{ s_{ik} + r_k + \beta \sum_{j=1}^{i-1} p_{kj} (\tilde{\Pi}V)_j + \beta \sum_{j=1}^{n} p_{kj} v_j \right\} . \quad (6.16)$$

Hastings' algorithm [18] is

$$V^{(n+1)} = (\Lambda \tilde{\Pi}) V^{(n)} \quad (6.17)$$

Kushner and Kleinman [30] use the standard "acceleration" [30] or "relaxation" [56] methods to define other policy improvement operators, such as "accelerated Jacobi," $\overset{a}{\Pi}$ where

$$(\overset{a}{\Pi}V)_i = \omega_i (\Pi V)_i + (1-\omega_i) v_i \quad (6.18)$$

and "accelerated Gauss-Seidel" $\overset{a}{\tilde{\Pi}}$ where

$$(\overset{a}{\tilde{\Pi}}V)_i = \omega_i (\tilde{\Pi}V)_i + (1-\omega_i) v_i \quad . \quad (6.19)$$

In (6.15) and (6.16) ω_i is a "relaxation factor" between 0 and 2. Kushner and Kleinman [30] report computational experience with the algorithms

$$V^{(n+1)} = \overset{a}{\Pi} V^{(n)} \quad (6.20)$$

$$V^{(n+1)} = \tilde{\Pi} V^{(n)} \quad (6.21)$$

$$V^{(n+1)} = \overset{a}{\tilde{\Pi}} V^{(n)} \quad (6.22)$$

as well as (6.12) and register the most satisfaction with (6.22).

Exactly analagous modifications* can be done for Λ, so the variety of possible algorithms, depending on version of Π, version of Λ, y and z, and the termination rule, is seemingly endless. A single equation which includes all these possibilities is

$$v^{(n+1)} = \left(\Lambda_{a_n}^{y_n} \; \Pi_{b_n}^{z_n} \right) v^{(n)} \tag{6.23}$$

where y_n and z_n are positive integers allowed to take on different values at each iteration, and where a_n and b_n are indicators for prescribing which version (if any) of Λ and Π to use at iteration n. A choice frequently made for y_n is $y_1=\infty, y_2=y_3=\ldots=0$. Under this option an initial "in-the-ballpark" set of values is obtained and subsequent values are found by policy improvement alone. Undoubtedly, as researchers in Markov decision processes get up-to-date with developments in numerical analysis, more algorithms will appear.

A final note on the methods motivated by numerical analysis is useful here. Totten [51] used the term *feedback* to describe the concept of using the most recently optained values $(\tilde{\Pi}v)_j$, $j=1,\ldots,i-1$, when computing $(\tilde{\Pi}v)_i$ in (6.10). We propose here to use the term *partial feedback* to describe operators that make use of some but not all of the most recent information. Let E be an N-tuple of integers with $0 \le e(i) \le i-1$, $i=1,\ldots,N$. The operator Π^E, where

---

* In fact, algorithm (6.22) with the operator Π replaced by Λ in the defining equation (6.19) is the well-studied successive overrelaxation method [56] for solving systems of simultaneous linear equations. Successive overrelaxation is applied in Chapter III, to value determination.

$$(\overset{E}{\Pi}V)_i = \min_{k=1,\ldots,N}\left\{s_{ik} + r_k + \beta \sum_{j=1}^{e(i)} p_{kj} (\overset{E}{\Pi}V)_j + \beta \sum_{j=e(i)+1}^{N} p_{kj} v_j\right\}, \quad (6.24)$$

is a *partial feedback policy improvement operator*. This operator is applied in Chapter III. When $e(i)=0$ for all i, partial feedback reduces to no feedback or Jacobi iteration. When $e(i)=i-1$ for all i, partial feedback is full feedback or Gauss-Seidel iteration.

6.2 Bounds and Action Elimination

Policy convergence means the repitition of a policy in two successive policy improvements. *Value convergence* means the repitition of values in two successive iterations, and hence the termination, of (6.23). It was mentioned previously that policy convergence is sufficient for value convergence in Howard's policy iteration method (6.10) and in Hastings' method (6.13), but not in general. Against the advantage of being able to terminate on policy convergence one must weigh the disadvantage of having to do a value determination, Λ^{∞}, i.e. solving an N by N system of linear equations, on each elucidated policy. The nature of this disadvantage is that, while the number of iterations is kept down, the time spent in each iteration is great, with most of that time spent in policy evaluation. There does not appear in the literature any study attempting to answer the question: when should numerous fast iterations be traded for few slow iterations, or vice versa?

If we choose to forego the advantage of terminating on policy convergence so that less time (or no time) is spent in policy evaluation, then we become interested in ways of reducing the time spent in policy

improvement. A strategy for doing this is to identify nonoptimal actions and to eliminate them from consideration in subsequent policy improvements. Implementation of this idea for Markov decision chains originated with MacQueen [31,32] and was extended by Porteus [35,37], Totten [51] and Hastings and Mello [20] to other stochastic decision processes.

The *MacQueen action elimination test* requires that an N-tuple lower bound V^L and an N-tuple upper bound V^U on the β-optimal values are available; i.e. V^L and V^U such that $v_i^L \le v_i^* \le v_i^U$, $i=1,\ldots,N$, are known. In terms of the standard formulation (given in Section 5) of the discounted Markov decision chain, the test is [32] : *Action* b *in state* i *is nonoptimal if*

$$r(i,b) + \beta \sum_{j=1}^{N} p(j|i,b)v_j^L > \min_{a\in A_i}\left[r(i,a) + \beta \sum_{j=1}^{N} p(j|i,a)v_j^U\right] . \quad (6.25)$$

The left side of the inequality gives the value state i would obtain if action b is used, assuming the *best* conceivable value for discounted future costs. The right side of (6.25) gives the value state i would obtain if some other action is used assuming the *worst* conceivable value for discounted future costs. This logic closely resembles that of fathoming in branch and bound integer programming.

In terms of the state-change formulation the MacQueen test is: *Deciding to change the state to* k *is nonoptimal in state* i *if*

$$s_{ik} + r_k + \beta \sum_{j=1}^{N} p_{kj}v_j^L > \min_{h}\left[s_{ih} + r_h + \beta \sum_{j=1}^{N} p_{hj}v_j^U\right] . \quad (6.26)$$

Note that the only state-dependent expressions in (6.26) are the state-change costs s_{ih}. Thus the MacQueen test requires much less computation when used on the state-change formulation than when used on the standard formulation.

MacQueen [31,32] generates sequences of lower and upper bounds, $\underline{V}^{(n)}$ and $\overline{V}^{(n)}$, which are used to eliminate actions at every iteration. The method is sometimes so discriminating, he reports, that all actions but one for a certain state are identified as nonoptimal, whence the optimal action is apparent. The bounds are computed as follows: $V^{(0)}$ is any N-tuple with $v_1^{(0)} = 0$. Then*

$$V^{(n+1)} = \Pi V^{(n)} - (\Pi V^{(n)})_1 \tag{6.27}$$

$$\underline{V}^{(n+1)} = V^{(n+1)} + \underline{\lambda}^{(n+1)} \tag{6.28}$$

$$\overline{V}^{(n+1)} = V^{(n+1)} + \overline{\lambda}^{(n+1)} \tag{6.29}$$

where

$$\underline{\lambda}^{(n+1)} = \left(\frac{1}{1-\beta}\right) \min_i \left[(\Pi V^{(n)})_i - v_i^{(n)}\right] \tag{6.30}$$

$$\overline{\lambda}^{(n+1)} = \left(\frac{1}{1-\beta}\right) \max_i \left[(\Pi V^{(n)})_i - v_i^{(n)}\right] . \tag{6.31}$$

MacQueen has shown that the sequence of bounds (6.28) and (6.29) converge monotonically from below and above, respectively, to V^*. [31]

---

* The interpretation of subtracting a scalar, $(\Pi V^{(n)})_1$, from an N-tuple, $\Pi V^{(n)}$, is that $v_i^{(n+1)} = (\Pi V^{(n)})_i - (\Pi V^{(n)})_1$ for all i.

6.3 Other Methods and Other Processes

If we choose not to forego the advantage of being able to terminate on policy convergence then decreasing the time required for value determination operations is the most important concern. To achieve this end methods using decomposition of the state space have been proposed by Wolfe and Dantzig [59], Kushner and Chen [29], Hartman and Lasdon [17], and Contreras [8]. MacQueen and Hitchcock [33] propose a method for value determination which is simply the degenerate case of (6.27)-(6.31) when there is only one action available at each state; in other words Π is replaced by Λ. MacQueen and Hitchcock present impressive computational experience, e.g., the upper bound (6.29) came within .01 per cent of the lower bound (6.28) after 5 iterations on a problem with 1000 states. Unfortunately these computational results are somewhat suspect due to the method they used for generating "random" transition probability matrices. They used four different types of transition matrices but they all suffered from the problem best demonstrated by the first. In the first type of matrix "the row entries were randomly generated by selecting for each entry a uniformly distributed random number between 0 and 1 and then normalizing by the row total." [33] The problem with this procedure is that the row entries will tend to be very small, all fairly near $\frac{1}{N}$. With $N=1000$ states the unnormalized sum of the row entries is going to be very close to 500 by the law of large numbers. Thus, the largest valued entry in the row can be no larger than about $\frac{1}{500}$, a rather restrictive condition. We cannot, therefore, be certain that MacQueen's and Hitchcock's [33] value determination method will perform well in general. Another method for value determination, *natural*

decomposition, applicable to non-ergodic chains, is introduced in Chapter III of this thesis.

Completing this survey of solution techniques for (finite-state, finite-action) Markov decision chains, there has recently appeared another methodology which differs significantly from the previously existing techniques. The idea of these methods, as proposed by Zaldivar and Hodgson [63] for undiscounted problems and by Young and Contreras [8] for the discounted case, is to attempt to forecast the final solution values V^* on the basis of early estimates $V^{(1)},V^{(2)},\ldots,V^{(n)}$ obtained from some iterative method of the form (6.23). This is still a fertile area for research as there are many options to take on: 1) the underlying iterative method, 2) the forecasting method, and 3) the decision rule on how often and when to forecast. Computational experience to date [8,63] is promising.

This section has surveyed the computational aspects of Markov decision chains. It is important to note that computational aspects (as well as applications) have been de-emphasized in the Markov decision process literature. A large portion of the literature is devoted to studying fundamental questions about more general stochastic decision processes wherein assumptions such as bounded costs, finite state spaces, finite action spaces, constant sojourn times in each state, and time-homogeneity, are relaxed. The status of this area of research is surveyed by Porteus [36].

6.4 Structured Policies

Another important area of research on stochastic decision processes is in the use of *structured policies* [36]. These are problem-specific

decision rules that are used to characterize optimal policies at much less computational expense than would be required by general approaches, and allow us (when they are optimal) to restrict attention to a smaller class of policies than would ordinarily be considered. The earliest of these is the familiar (s,S) inventory policy [21,p.503] which applies to the example of Section 2.3 under certain assumptions on holding, shortage and replenishment costs.

Derman [12] proposed a simple heuristic* for certain machine replacement problems. He described a unit as being in one of $N+1$ states with state 0 meaning inoperative and states $1,2,\ldots,N$ referring to progressively better conditions. He defines "a *control limit policy* as one which always replaces the unit whenever the observed state is $i_0, i_0+1, \ldots, N$, and never replaces the unit in states $0,1,\ldots,i_0-1$; state i_0 is the *control limit*. Under certain conditions on the Markov transition probabilities $\{p_{ij}\}$ there always exists a control limit policy that is optimal." [12,p.122] Derman's sufficient conditions for optimality of a control limit policy are equivalent to:

$$i > j \Rightarrow \sum_{h=0}^{k} p_{ih} \leq \sum_{h=0}^{k} p_{jh} \quad \text{, for all } i,j,k \text{ .} \qquad (6.32)$$

An intuitive interpretation of (6.32) is: i is a "healthier" state than j, therefore the machine has a smaller probability of moving from i than from j to one of the "sick" states $0,1,\ldots,k-1$. This intuition

---

* "Heuristic" is used here in the sense of "an idea motivated by common sense and serving as a guide to solution," not in the sense of "necessarily inexact" which has crept into usage among operations researchers.

best applies when $p_{ij}=0$ for $i<j$, that is when the machine cannot improve its condition between inspections.

Miller [34] employed a heuristic dispatching rule to minimize long run average expected back orders in a military equipment repair system. Miller showed that the policy obtained from the heuristic rule was optimal for the Markov decision process model of the dispatching problem.

Swersey [49] formulated the problem of deciding how many fire-fighting units to dispatch to a fire alarm as a semi-Markov decision process. A simple decision rule based on a cutoff value for an estimate of the seriousness of the fire was shown to generate an optimal policy.

Stochastic decision processes are applicable to the optimal control of queueing systems [38] where, for instance, the number of servers [3,4] or the rate of service [45] is a dynamic decision variable. Structured policies such as turning on (or leaving on) a removable server when k or more customers are present, and turning off the server when it is idle, have been investigated and in some cases shown optimal [3].

Specially structured policies which are characterized by a single "cutoff" value of a parameter, such as a control limit or k customers in the queue or probability s^* that a fire is serious [49], belong to the class of policies called *monotone policies* and studied by Serfozo [45]. Experiments with monotone policies are described in this thesis for some stochastic decision processes in location analysis.

CHAPTER III

A SINGLE-FACILITY DYNAMIC PROBABILISTIC LOCATION PROBLEM

In this chapter the methodology of state-change Markov decision chains is applied to a dynamic probabilistic location problem involving a single new facility and a single existing facility. The next chapter extends consideration to dynamic probabilistic location problems with multiple existing facilities.

Conforming in this work to the literature of location analysis, the distinction between a *new* facility and an *existing* facility is that the location of a *new* facility is under a decision maker's control and the location of an *existing* facility is not under a decision maker's control. The new facility may be thought of as a service provider (e.g., ambulance, warehouse, lift truck) or *server* and the existing facility may be thought of as a service requester (e.g., accident victim, market area, movement requirement) or *customer*.

The dynamic aspect of the problem introduced here is that facilities (new or existing) may change locations over time. In each time period, certain costs are incurred and the location (or relocation) decision has to be made again for the new facility. The probabilistic aspect of the problem is that the changing location of the existing facility is considered to be a random phenomenon. A great variety of problems may be formulated within this framework through varying assumptions about the random phenomena and the costs. One particular formulation is

analyzed in detail for the remainder of this chapter.

1. The Process

The location of both the new and the existing facility, while changing over time, are confined to a finite set $N = \{1,\ldots,N\}$ of possible locations. The changing location of the existing facility is described by a known Markov transition matrix P while the location of the new facility depends on the instructions of a decision maker. Costs are incurred in two ways: 1) when the new facility is relocated by choice, and 2) when the two facilities interact for service in each time period.

The state of the system at time t, $t=0,1,2,\ldots$, is represented as an ordered pair $(X_t,A_t) \in N \times N$, where X_t is the location of the new facility and A_t is the location of the existing facility. An arbitrary realization of the two-dimensional state variable is written as (i,j); the next realization is (k,ℓ). The dual nature of the transition is expressed by the diagram of Figure 5.

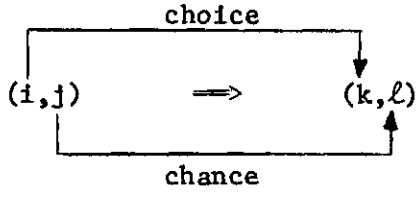


Figure 5. Typical Transition

The data assumed to be available for this problem are:

P = transition matrix for existing facility location, $N \times N$

F = matrix of relocation costs, $N \times N$

F' = matrix of service costs, $N \times N$

β = discount factor.

Both types of costs, new facility relocation and facility interaction, are location-dependent but not time-dependent. f_{ij} is the cost incurred whenever the new facility is moved from i to j. f'_{ij} is the cost of service whenever the new and existing facilities are at i and j, respectively. The order of events is:

1) the decision-maker observes (X_{t-1}, A_{t-1}) and chooses X_t,
2) the relocation cost $f(X_{t-1}, X_t)$ is incurred,
3) the chance location A_t is realized and
4) the cost $f'(X_t, A_t)$ is incurred;

then the process repeats. The objective is to minimize the expected present worth of all the costs, i.e.,

$$\text{Minimize} \quad E\left[\sum_{t=1}^{\infty} \left(f(X_{t-1}, X_t) + f'(X_t, A_t)\right)\beta^{t-1}\right] \quad . \qquad (1.1)$$

With respect to our ability to solve this type of problem there is no loss of generality in assuming that all costs are proportional to distance. Analysis of the problem is presented with F given as a distance matrix, in units such that the cost of service when the two facilities are separated by a distance f_{ij} is f_{ij}, and the cost of relocating a distance f_{ij} is αf_{ij}. α is referred to as the *relocation cost factor*. This cost convention is used because it reduces the data requirements; our methods apply, however, when F and F' have any desired form.

2. Modeling the Location Problem as a Two-Dimensional State-Change Markov Decision Chain

The problem is modeled as a two-dimensional state-change Markov decision chain because the state is identified as an ordered pair. The decision variables are trebly subscripted, for $i,j,k \in N$

$$d_{ijk} = 1 \text{ , if the decision-maker chooses to move the new facility to location } k \text{ whenever the new facility is at location } i \text{ and the existing facility is at location } j \text{ ,}$$

$$= 0 \text{ , otherwise.}$$

When the decision $d_{ijk} = 1$ is implemented the state is changed from (i,j) to (k,j). To completely conform with the state-change methodology, decision variables should be quadruply subscripted as $d_{ij,k\ell}$, where $d_{ij,k\ell}$ is the probability of choosing to change the state from (i,j) to (k,ℓ). Of course, the simplifying fact

$$d_{ij,k\ell} = \begin{cases} d_{ij,kj} = d_{ijk} & \text{, if } \ell = j \\ 0 & \text{, if } \ell \neq j \end{cases} \qquad \text{for all } i,j,k,\ell \in N \qquad (2.1)$$

i.e., that the location of the existing facility cannot be changed by choice, obviates the four-dimensional variables; but this notation will be used briefly for expository purposes to describe the complete transition matrix of the state-change process.

2.1 The Complete Transition Matrix

When the existing facility changes its location from location j to location ℓ, which occurs with conditional probability $p_{j\ell}$, the state

is changed from (i,j) to (i,ℓ). Thus, chance-determined changes of state are characterized by probabilities $p_{ij,k\ell}$ where

$$p_{ij,k\ell} = \begin{cases} p_{ij,i\ell} = p_{j\ell}, & \text{if } k=i \\ 0, & \text{if } k\neq i \end{cases} \quad \text{for all } i,j,k,\ell \in N . \quad (2.2)$$

When the probabilities $d_{ij,k\ell}$ and $p_{ij,k\ell}$ are placed in two-dimensional arrays $\overline{D}$ and $\overline{P}$, respectively, with a row and column for each state, the complete transition matrix T is determined* by $T = \overline{DP}$. As an example let $N=3$ and let

$$P = \begin{pmatrix} .1 & .7 & .2 \\ .4 & 0 & .5 \\ .6 & .1 & .3 \end{pmatrix} .$$

Then

$$\overline{D} = \begin{array}{c} \\ 11 \\ 12 \\ 13 \\ 21 \\ 22 \\ 23 \\ 31 \\ 32 \\ 33 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \end{array} \\ \begin{pmatrix} d_{111} & & & d_{112} & & & d_{113} & & \\ & d_{121} & & & d_{122} & & & d_{123} & \\ & & d_{131} & & & d_{132} & & & d_{133} \\ d_{211} & & & d_{212} & & & d_{213} & & \\ & d_{221} & & & d_{222} & & & d_{223} & \\ & & d_{231} & & & d_{232} & & & d_{233} \\ d_{311} & & & d_{312} & & & d_{313} & & \\ & d_{321} & & & d_{322} & & & d_{323} & \\ & & d_{331} & & & d_{332} & & & d_{333} \end{pmatrix} \end{array}$$

---

* The same ordering of states must be used for rows and columns of each matrix, of course.

$$\overline{P} = \begin{array}{c} \\ 11 \\ 12 \\ 13 \\ 21 \\ 22 \\ 23 \\ 31 \\ 32 \\ 33 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \end{array} \\ \begin{pmatrix} .1 & .7 & .2 & & & & & & \\ .4 & 0 & .5 & & & & & & \\ .6 & .1 & .3 & & & & & & \\ & & & .1 & .7 & .2 & & & \\ & & & .4 & 0 & .5 & & & \\ & & & .6 & .1 & .3 & & & \\ & & & & & & .1 & .7 & .2 \\ & & & & & & .4 & 0 & .5 \\ & & & & & & .6 & .1 & .3 \end{pmatrix} \end{array}$$

and

$$T = \begin{array}{c} \\ 11 \\ 12 \\ 13 \\ 21 \\ 22 \\ 23 \\ 31 \\ 32 \\ 33 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \end{array} \\ \begin{pmatrix} .1d_{111} & .7d_{111} & .2d_{111} & .1d_{112} & .7d_{112} & .2d_{112} & .1d_{113} & .7d_{113} & .2d_{113} \\ .4d_{121} & 0 & .5d_{121} & .4d_{122} & 0 & .5d_{122} & .4d_{123} & 0 & .5d_{123} \\ .6d_{131} & .1d_{131} & .3d_{131} & .6d_{132} & .1d_{132} & .3d_{132} & .6d_{133} & .1d_{133} & .3d_{133} \\ .1d_{211} & .7d_{211} & .2d_{211} & .1d_{212} & .7d_{212} & .2d_{212} & .1d_{213} & .7d_{213} & .2d_{213} \\ .4d_{221} & 0 & .5d_{221} & .4d_{222} & 0 & .5d_{222} & .4d_{223} & 0 & .5d_{223} \\ .6d_{231} & .1d_{231} & .3d_{231} & .6d_{232} & .1d_{232} & .3d_{232} & .6d_{233} & .1d_{233} & .3d_{233} \\ .1d_{311} & .7d_{311} & .2d_{311} & .1d_{312} & .7d_{312} & .2d_{312} & .1d_{313} & .7d_{313} & .2d_{313} \\ .4d_{321} & 0 & .5d_{321} & .4d_{322} & 0 & .5d_{322} & .4d_{323} & 0 & .5d_{323} \\ .6d_{331} & .1d_{331} & .3d_{331} & .6d_{332} & .1d_{332} & .3d_{332} & .6d_{333} & .1d_{333} & .3d_{333} \end{pmatrix} \end{array}$$

$$= \overline{D}\,\overline{P} \qquad (2.3)$$

The sums

$$t_{ij,k\ell} = \sum_{q,r} (d_{ij,qr})(p_{qr,k\ell}) \tag{2.4}$$

which comprise the elements of the matrix product $\overline{DP}$ have at most one nonzero term. This agrees with the argument that the complete probability of the transition $(i,j) \to (k,\ell)$ is the joint probability that the location of the new facility changes by *choice* from node i to node k and the location of the existing facility changes by *chance* from node j to node ℓ. Since these events occur independently

$$\begin{aligned} t_{ij,k\ell} &= \Pr\{X_{n+1}=k, A_{n+1}=\ell \mid X_n=i, A_n=j\} \\ &= \Pr\{X_{n+1}=k \mid X_n=i, A_n=j\} \Pr\{A_{n+1}=\ell \mid X_n=i, A_n=j\} \\ &= d_{ijk}\, p_{j\ell} \quad . \end{aligned} \tag{2.5}$$

A complete transition is characterized by the diagram of Figure 6.

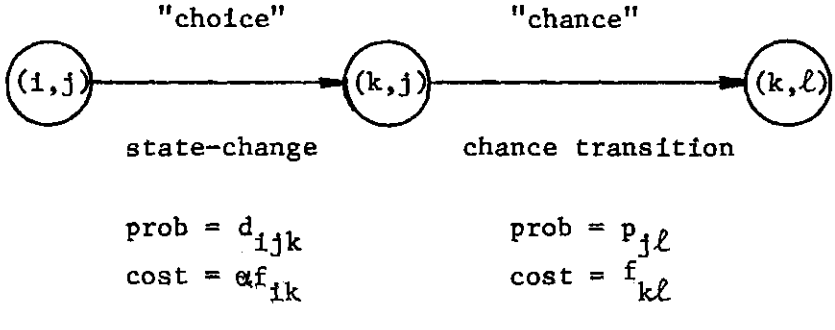


Figure 6. Complete Transition for Single-Facility Problem

When the states are ordered $(1,1),(1,2),\ldots,(1,N),(2,1),\ldots,(N,1),(N,2)\ldots,(N,N)$, as in the preceeding example (and as in all subsequent developments), the N^4 elements of the T matrix may be partioned into N^2 submatrices, T_{ik}, $i,k=1,\ldots,N$. Then $T =$

	11	12 ... 1N	21 ... 2N	. . .	N1 ... NN
11					
12 : : 1N		T_{11}	T_{12}	. . .	T_{1N}
21 : : 2N		T_{21}	T_{22}	. . .	T_{2N}
. . .		. . .	. . .	. . . . . .	. . .
N1 : : NN		T_{N1}	T_{N2}	. . .	T_{NN}

(2.6)

where T_{ik} consists of the probabilities of all transitions, $(i,j) \to (k,\ell)$, $j,\ell = 1,\ldots,N$, from states having the new facility in node i to states having the new facility in node k. Under any pure, stationary state-change policy the complete transition matrix, when written in form (2.6), has a special structure:

1) every row of T has all of its nonzero elements concentrated in one of the partitions T_{ik}; and (2.7)

2) if the j^{th} row of T_{ik} is not all zero then it is the same as the j^{th} row of P. (2.8)

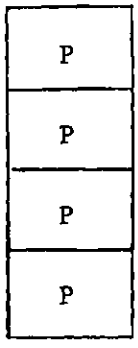
Let the state-change policy given by the following table be used in the preceeding example.

State, (i,j)	d_{ijk} for k = 1	2	3
11	0	1	0
12	1	0	0
13	0	1	0
21	0	0	1
22	0	0	1
23	1	0	0
31	1	0	0
32	0	1	0
33	0	0	1

Then

$$T = \begin{array}{c} \\ 11 \\ 12 \\ 13 \\ 21 \\ 22 \\ 23 \\ 31 \\ 32 \\ 33 \end{array} \begin{array}{c} \begin{array}{ccccccccc} 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \end{array} \\ \left(\begin{array}{ccccccccc} & & & .1 & .7 & .3 & & & \\ .4 & 0 & .6 & & & & & & \\ & & & .6 & .1 & .3 & & & \\ & & & & & & .1 & .7 & .3 \\ & & & & & & .4 & 0 & .6 \\ .6 & .1 & .3 & & & & & & \\ .1 & .7 & .3 & & & & & & \\ & & & .4 & 0 & .6 & & & \\ & & & & & & .6 & .1 & .3 \end{array} \right) \end{array} \qquad (2.9)$$

To illustrate property (2.7) in (2.9), note that the positive elements of row (2,3) are all in the submatrix T_{21} ; this is because $d_{231}=1$. Further, the elements $t_{23,11}$, $t_{23,12}$, $t_{23,13}$ are respectively, p_{31} , p_{32} , p_{33} . demonstrating property (2.8). Heuristically, the special structure of T may be thought of as evolving as follows: N copies of the P matrix are "stacked" on top of each other and then the rows of the "stack" are individually "slid" an integer multiple of N places to the right or left. See Figure 7.



		Row 1	
Row 2			
			Row 3
	Row 4		
	Row 1		
		Row 2	
		Row 3	
Row 4			
Row 1			
			Row 2
	Row 3		
	Row 4		
		Row 1	
			Row 2
	Row 3		
Row 4			

Figure 7. Structure of T

Figure 7 is an example with N=4 where Row i means the i^{th} row of P .

A final point on T 's structure: the density of T (proportion of elements that are nonzero) equals the density of P divided by N . With density no greater than $\frac{1}{N}$, T is an extremely sparse matrix.

2.2 The Value Determination Equations and the Extremal Equations

This section develops two things necessary for analyzing the location problems: 1) the mechanism for evaluating a given state-change policy (analagous to equations (2.2) and (2.3) of Chapter II), and 2) the extremal equations for characterizing a β-optimal policy (like (4.1) of Chapter II).

The transition probabilities $t_{ij,k\ell}$ associated with a given state change policy, $D=\{d_{ijk}\}$, are given by (2.5). The associated transition costs are the next item investigated.

Consider the expected value of the costs incurred in a single period with the process beginning in state (i,j) . This quantity is written r_{ij} . By convention a time period begins when it is the decision maker's turn to change the state. If he changes the state to (k,j) by moving the new facility to location $k\varepsilon N$, then the expected single period cost is the expected cost of the chance transition from (k,j) , given by $\sum_{\ell=1}^{N} p_{j\ell} f_{k\ell}$, plus the cost of changing the state from (i,j) to (k,j) , αf_{ik} . Since the change from (i,j) to (k,j) is made with probability d_{ijk} ,

$$r_{ij} = \sum_{k\varepsilon N} d_{ijk}(\alpha f_{ik} + \sum_{\ell=1}^{N} p_{j\ell} f_{k\ell}) \qquad (2.10)$$

for all $(i,j)\in N\times N$. This is simply equation (2.1) of Chapter II for the particular state-change Markov decision chain considered in this chapter. Equation (2.10) is illustrated by Figure 8.

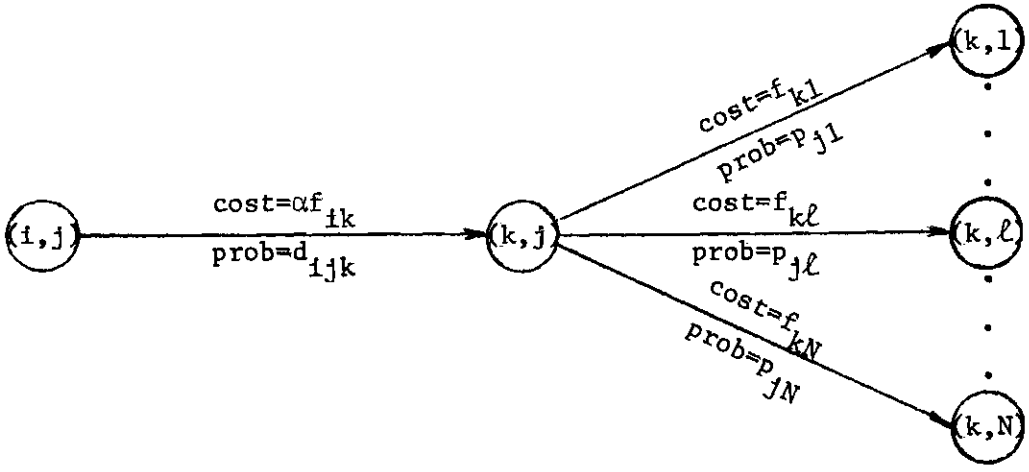


Figure 8. Illustration of the
Expected Immediate Cost Equation (2.10)

The inner summation of (2.10) is an important quantity, whose symbol will be

$$w_{jk} = \sum_{\ell=1}^{N} p_{j\ell} f_{k\ell} \quad ; \tag{2.11}$$

note, w_{jk} is the expected value of the immediate cost of service when the existing facility is at location j and the new facility has just been moved to location k . The matrix

$$W = P F^{T} \tag{2.12}$$

may be computed and tabulated at the outset before specific policies are selected or evaluated.

Next consider the state values, v_{ij} . These must satisfy the value equations (1.3) of Chapter II, which, for the case of a two-dimensional state, have the form

$$v_{ij} = r_{ij} + \sum_{k\in N} \sum_{\ell\in N} t_{ij,k\ell}\, v_{k\ell} \quad , (i,j)\in N\times N \quad . \tag{2.13}$$

Substituting

$$r_{ij} = \sum_{k\in N} d_{ijk}\left(\alpha f_{ik} + w_{jk}\right) \tag{2.14}$$

and

$$t_{ij,k\ell} = d_{ijk}\, p_{j\ell} \tag{2.5}$$

in (2.13) yields

$$v_{ij} = \sum_{k\in N} d_{ijk}\left[\alpha f_{ik} + w_{jk} + \beta \sum_{\ell\in N} p_{j\ell} v_{k\ell}\right] \tag{2.15}$$

for all $(i,j)\in N\times N$. Equations (2.15) were derived in the same manner as equations (2.2) of Chapter II and they serve the same purpose. Solving (2.15) is the value determination operation for any stationary state-change policy, $\{d_{ijk}\}$, in the single-facility location problem.

Finally, in this section we specialize the value determination equations for the case of pure policies and use the result to motivate the extremal equations. Let $k_{ij}\in N$ be the location to which the new facility is moved whenever state (i,j) is observed. A pure state-change policy is denoted by $K = \{k_{ij}\}$. The relationship between the notation

$D = \{d_{ijk}\}$ and K, in the case of stationary pure policies, is

$$d_{ijk} = \begin{cases} 1 & , \text{ if } k=k_{ij} \\ 0 & , \text{ otherwise} \end{cases} \quad , i,j,k \epsilon N \quad . \tag{2.16}$$

Using K, (2.15) reduces to

$$v_{ij} = \alpha f(i,k_{ij}) + w(j,k_{ij}) + \beta \sum_{\ell \epsilon N} p_{j\ell} v(k_{ij},\ell) \tag{2.17}$$

for all $(i,j) \epsilon N \times N$. Equations (2.17) are intuitive; the three terms of the right-hand side are, respectively, the relocation (state-change) cost, the expected service (state-visit) cost and the expected discounted future costs, all of which depend on the current pure policy.

The extremal equations for the single-facility location problem have the structure of (2.17) where the specified decision k_{ij} is replaced by an arbitrary location k, and the objective is to choose k's which minimize the state values, i.e.

$$v_{ij} = \min_{k \epsilon N} \left(\alpha f_{ik} + w_{jk} + \beta \sum_{\ell \epsilon N} p_{j\ell} v_{k\ell} \right) \quad , (i,j) \epsilon N \times N \quad . \tag{2.18}$$

We have now represented the problem to be solved: find a pure policy $K = \{k_{ij}\}$ such that the values associated (via (2.17)) with K, v_{ij}, are optimal in the sense of (2.18). The objective originally stated, (1.1), is equivalent to v_{ij} when the condition $(X_0,A_0) = (i,j)$ is assumed. The remainder of the chapter is devoted to solving (2.18).

If different functions F and F' are used to describe relocation and service costs respectively, then equations (2.18) are still appropri-

ate for solving the problem

$$\text{Minimize } E\left\{\sum_{t=1}^{\infty}[f(X_{t-1},X_t) + f'(X_t,A_t)]\beta^{t-1}\right\}$$

with $\alpha=1$ and $w_{jk} = \sum_{\ell=1}^{N} p_{j\ell} f'_{k\ell}$.

3. Solution Techniques

All the known methods for solving Markov decision chains, including those reported in Chapter II, are applicable to solving the two-dimensional, state-change, location problem of this chapter. It would be unwise, however, to apply those techniques without first investigating the special characteristics of our particular problem. This section describes the special characteristics and shows how they can be exploited in each component of the solution techniques.

The remainder of this chapter, devoted entirely to solving the problem introduced in Sections 1 and 2, is organized as follows: First some notation is introduced which enables us to describe the value determination operation (2.17) as a matrix equation. Section 3.1 considers initializing the policy and the values. The state values associated with a certain initial policy, called the *hypermyopic policy*, are obtained at little computational effort. Section 3.2 discusses various methods for solving the value determination equations, including the iterative methods mentioned in Chapter II. An unusual phenomenon of a nonoptimal policy having some of its associated values at their optima is also discussed in Section 3.2 and so is a new algorithm for value determination on nonergodic chains. Section 3.3 illustrates some alternative policy

improvement operators based on the partial feedback concept introduced in Chapter II. Section 3.4 applies the MacQueen action elimination test to the location problem. Finally, computational experience is reported in Section 4 for several algorithms incorporating various options on the procedures in Section 3.

In order to describe the subsequent developments compactly, the following notation is employed:

V = $N\times N$ matrix of v_{ij} .

$\hat{V}$ = $N^2\times 1$ vector of v_{ij} in the usual order, $v_{11},v_{12},\ldots,v_{1N}$, $v_{21},\ldots,v_{2N},\ldots,v_{N1},\ldots,v_{NN}$.

$\hat{V}_i$ = $N\times 1$ vector of $v_{i1},\ldots,v_{iN}$.

R = Current $N\times N$ matrix of expected immediate costs,

$$r_{ij} = \alpha f(i,k_{ij}) + w(j,k_{ij}) \tag{3.1}$$

$\hat{R}$ = $N^2\times 1$ vector of r_{ij}

$\hat{R}_i$ = $N\times 1$ vector of $r_{i1},\ldots,r_{iN}$.

The system of value determination equations (2.17) for evaluating a pure state-change policy K is equivalent to

$$\hat{V} = \hat{R} + \beta T\hat{V} \tag{3.2}$$

or

$$Q\hat{V} = \hat{R} \tag{3.3}$$

where

$$Q = I - \beta T \ , \tag{3.4}$$

I is an $N^2\times N^2$ identity matrix, and T is the complete transition matrix with elements given by (2.5). Note, $\hat{R}$ and T (and consequently

$\hat{V}$) are dependent on the current policy K . The dependence of $\hat{R}$ on K is given by (3.1). The dependence of T on K is given by (2.5) and (2.16), or it may be expressed by a more precise statement of property (2.7):

> Row (i,j) of T , the complete transition matrix for the two-dimensional state-change process operating under the pure policy K , has all its nonzero elements concentrated in the submatrix $T(i,k_{ij})$.

3.1 Initialization

The amount of work required to initialize an iterative procedure for solving Markov decision chains is dependent on both the solution procedure selected and the structure of the particular problem. When using an algorithm such as Howard's [22] or Hastings' [18] where value determination operations (VDO's) are performed, the initial VDO is generally as costly as subsequent VDO 's . The policy most often used for the initial VDO is the *myopic* (or *greedy*) *policy*, that policy which minimizes expected immediate costs. When using successive approximation type methods [30,57], which neglect VDO's , the myopic policy is still the most common initial policy. In particular, when a nonfeedback (Jacobi) policy improvement routine is used to generate the initial policy and the values of states are initially set to zero, the myopic policy will precipitate.

3.1.1 Myopic and Hypermyopic Policies. For the location problem the myopic policy instructs the new facility to move to the location which minimizes the sum of relocation cost and the expected value of the

next service cost. Let

$$r_{ij}(k) = \alpha f_{ik} + w_{jk} \quad , (k,j)\epsilon N\times N \quad . \tag{3.5}$$

Then the myopic policy is $K^* = \{k^*_{ij}\}$ where

$$r_{ij}(k^*_{ij}) = \min_{k\epsilon N} r_{ij}(k) \quad , (i,j)\epsilon N\times N \quad . \tag{3.6}$$

This policy is easy to identify, and its use initially would be in concord with accepted practice; yet, there is another policy which has more advantages.

An alternative initial policy is given the name here of *hypermyopic*. The name conveys the idea that the hypermyopic policy exaggerates the myopic tendency of considering only the near-term. Precisely, the hypermyopic policy instructs the new facility to choose the location which minimizes relocation cost alone. It is assumed quite naturally that $f_{ii}=0$ for all $i\epsilon N$, i.e. there is no relocation cost when the new facility does not move; so the hypermyopic policy is simply to leave the new facility at its current location, or $k_{ij}=i$ for all (i,j). The advantages of the hypermyopic policy are that it is a reasonable starting policy obtainable without any calculation, and, much more importantly, that it can be evaluated with far less effort than is required for usual VDO 's . The latter advantage is described in detail in the next section.

In certain instances the myopic and the hypermyopic policies are identical. If this is known to be the case, then computation of (3.6) is an unnecessary expense. Sufficient conditions for equivalence of the two

policies are:

$$f_{i\ell} \leq f_{ik} + f_{k\ell} \quad \text{for all } i,\ell,k \in N \ . \tag{3.7}$$

and

$$\alpha \geq 1 \ . \tag{3.8}$$

To verify the sufficiency, note, for all $i,j,k \in N$,

$$\begin{aligned} r_{ij}(k) &= \alpha f_{ik} + w_{jk} \\ &= \alpha f_{ik} + \sum_{\ell} p_{j\ell} f_{k\ell} \\ &= \sum_{\ell} p_{j\ell} (\alpha f_{ik} + f_{k\ell}) \\ &\geq \sum_{\ell} p_{j\ell} f_{i\ell} \qquad \text{[by (3.7) and (3.8)]} \\ &= w_{ji} \\ &= r_{ij}(i) \quad ; \end{aligned}$$

whence $k^*_{ij}=i$, i.e. the myopic policy is hypermyopic when (3.7) and (3.8) are assumed.

Of the two sufficient conditions, (3.7) is by far the less restrictive. If the entries f_{ij} are obtained by measuring the distance between locations i and j with respect to a metric; then, by definition (of metric), the triangle inequality, (3.7), must hold. Since some metric (e.g. rectilinear or Euclidean) is probably going to be used to measure distance in every practical case, (3.7) is a reasonable assumption.

On the other hand, condition (3.8) may or may not hold in cases of practical interest. Whether or not $\alpha \geq 1$ depends on the nature of the service activity whose cost is modeled by f_{ij}. If the existing facility moves to the new facility when they interact for service and if the new facility is the more difficult to move, then $\alpha > 1$ is appropriate. If service involves a round trip taken by the new facility and if moving for service and moving to relocate cost about the same per mile, then $\alpha \approx \frac{1}{2}$. For a case when α is very much less than one, consider any case in which the service interactions involve movement of tangible or intangible *arms* of the facilities, such as when an office copier serves offices and only people and paper actually move.

The sufficient conditions (3.7) and (3.8) may be weakened without destroying the equivalence of the myopic and hypermyopic policies. The inequality

$$\alpha f_{ik} + f_{k\ell} \geq f_{i\ell} \quad , \text{ for all } i,k,\ell \in N \ ,$$

required in the validation of the sufficient conditions above, can possibly hold when $\alpha < 1$. Even in situations when the two policies are not identical, experience indicates that the hypermyopic is the preferred initial policy. The reason for this is demonstrated below.

3.1.2 Obtaining Initial Values With the Hypermyopic Policy. A key issue in the design and implementation of iterative algorithms is the tradeoff between quality of the initial solution and the cost of obtaining it. The state values associated with the hypermyopic policy are a reasonable first estimate of the solution to (2.18); in fact some of

these values may turn out to be optimal. (Section 3.2.2) Furthermore, determination of the state values for the hypermyopic policy can be accomplished at the cost of inverting an N by N matrix, a remarkably low cost considering that ordinarily the value determination operation ((2.17) or (3.2)) is a system of N^2 equations in N^2 unknowns.

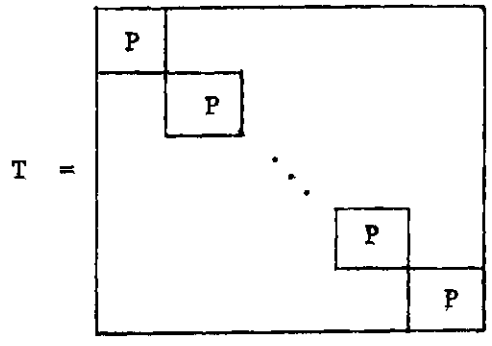
The hypermyopic policy is $k_{ij}=i$ for all $(i,j)\in N\times N$; or equivalently,

$$d_{ijk} = \begin{cases} 1 & \text{if } k=i \\ 0 & \text{otherwise} \end{cases} \quad , \; i,j,k\in N \; .$$

This results in complete transition probabilities

$$t_{ij,k\ell} = d_{ijk}\, p_{j\ell} = \begin{cases} p_{j\ell} & \text{if } k=i \\ 0 & \text{otherwise} \end{cases} \quad , \; i,j,k,\ell\in N \; ; \quad (3.9)$$

and the partitioned form of T , (2.6), is block-diagonal with $T_{ii}=P$ for all $i\in N$; or



Consequently, the value determination operation (3.2) for the hypermyopic policy is

$$\begin{bmatrix} \hat{V}_1 \\ \hat{V}_2 \\ \vdots \\ \hat{V}_N \end{bmatrix} = \begin{bmatrix} \hat{R}_1 \\ \hat{R}_2 \\ \vdots \\ \hat{R}_N \end{bmatrix} + \beta \begin{bmatrix} P & & & \\ & P & & \\ & & \ddots & \\ & & & P \end{bmatrix} \begin{bmatrix} \hat{V}_1 \\ \hat{V}_2 \\ \vdots \\ \hat{V}_N \end{bmatrix} \tag{3.10}$$

which decomposes to

$$\hat{V}_i = \hat{R}_i + \beta P \hat{V}_i \quad , \; i=1,\ldots,N \quad . \tag{3.11}$$

Since $f_{ii}=0$, the immediate cost vector $\hat{R}_i$ is

$$\hat{W}_i = (w_{1i},w_{2i},\ldots,w_{Ni})^{\tau} = (\sum_{\ell} p_{1\ell} f_{i\ell},\ldots,\sum_{\ell} p_{N\ell} f_{i\ell})^{\tau} \quad .$$

Hence,

$$\hat{V}_i = \hat{W}_i + \beta P \hat{V}_i \quad , \; i=1,\ldots,N \quad . \tag{3.12}$$

Then, expressing all the vector equations (3.12) as a single matrix equation,

$$V^{\tau} = W + \beta P V^{\tau} \quad . \tag{3.13}$$

In terms of the original data, this equation, which the state values must satisfy under the hypermyopic policy, is

$$V^{\tau} = PF^{\tau} + \beta PV^{\tau}$$

or

$$V^{\tau} = (I-\beta P)^{-1} PF^{\tau} \quad . \tag{3.14}$$

Note that in (3.14) I is N by N whereas in (3.4) I is N^2 by N^2. Thus, an initial solution for the state values can be obtained by inverting a matrix of order N rather than N^2.

3.2 Methods of Value Determination

When policies other than the hypermyopic are used, a value determination operation is considerably more difficult. Thus, we are motivated to use the values obtained from the initial VDO as "in-the-ballpark" estimates and then proceed with an iteration scheme that makes infrequent use of VDO 's . For those infrequent occasions when a VDO is needed, such as when policy convergence but not value convergence occurs in an algorithm of the form of Figure 4 in Chapter 2, a good value determination method is required. This section investigates alternative methods.

3.2.1 Iterative Methods. As indicated in Chapter II the iterative techniques of numerical analysis have been applied to Markov decision chains. The solution of the value determination equation

$$\hat{V} = \hat{R} + \beta T\hat{V} \tag{3.2}$$

for the location problem lends itself especially to iterative methods. Applying the simplest iterative VDO that has been used in Markov decision chains [33], we may solve (3.2) as

$$\hat{V}^{(n+1)} = \hat{R} + \beta T \hat{V}^{(n)} \quad . \tag{3.15}$$

Introducing the feedback concept, (3.15) may be modified to

$$v_{ij}^{(n+1)} = \tilde{v}_{ij}^{(n+1)} = \begin{cases} r_{ij} + \beta \sum_{\ell=1}^{N} p_{j\ell} v^{(n+1)}(k_{ij},\ell) & , \text{ if } i>k_{ij} \\ r_{ij} + \beta \sum_{\ell=1}^{j-1} p_{j\ell} v^{(n+1)}(k_{ij},\ell) + \beta \sum_{\ell=j}^{N} p_{j\ell} v^{(n)}(k_{ij},\ell) & , \text{ if } i=k_{ij} \\ r_{ij} + \beta \sum_{\ell=1}^{N} p_{j\ell} v^{(n)}(k_{ij},\ell) & , \text{ if } i<k_{ij} \end{cases} \tag{3.16}$$

This may be further modified using overrelaxation to

$$v_{ij}^{(n+1)} = \omega \tilde{v}_{ij}^{(n+1)} + (1-\omega) v_{ij}^{(n)} \quad . \tag{3.17}$$

Equation (3.17) is essentially the successive overrelaxation method [56] (it differs slightly when $k(i,j)=i=j$) which is generally acknowledged as the best of the iterative methods when applicable.*

A sufficient condition [56] for the convergence of successive overrelaxation, when solving for x in the square system $Ax=b$, is that

$$|a_{ii}| - \sum_{\substack{j=1 \\ j\neq i}}^{N} |a_{ij}| > 0 \quad , i=1,\ldots,N \quad . \tag{3.18}$$

---

* With the substantial improvements in computer storage and the development of special purpose file structures that have appeared since the iterative methods were proposed, the sentiment is growing among mathematicians that direct (Gaussian elimination type) methods are generally preferred. In the present context iterative methods are considered because they can easily exploit the special structure of T in (3.2). [64]

The system $Q\hat{V} = \hat{R}$ where $Q = I-\beta T$ satisfies the condition (3.18) of *diagonal dominance*. The diagonal element of row (i,j) in Q is $1-\beta t_{ij,ij} = 1-\beta d_{iji}p_{jj}$ by (2.5). The off-diagonal elements for that row sum to $-\beta(1-d_{iji}p_{jj})$ since T is a stochastic matrix. Diagonal dominance in Q is assured since $|1-\beta d_{iji}p_{jj}| - |-\beta(1-d_{iji}p_{jj})| = 1-\beta > 0$.

A more important reason than diagonal dominance for the attractiveness of successive overrelaxation in solving (3.2) is that the special structure of T is exploited in (3.16). There are only N terms, not N^2 terms, in the summations of (3.16). Thus, to apply successive overrelaxation we do not have to store the complete T matrix. We only have to store P , K , V and R instead of T , V , and R , a requirement of $4N^2$ words instead of N^4+2N^2 .

3.2.2 Simplifying the Value Determination Operation Through Prior Knowledge of Certain State Values. For experimental purposes several single-facility problems were solved using Howard's policy iteration method and Hastings' modification of policy iteration (equations (6.10) and (6.17) of Chapter II). Thus, we have observed changes in the state values when VDO 's are performed for every specified policy in an iterative procedure. Experience indicated that recurrence of state values from one VDO to the next for some of the states is a common occurence. An explanation of the reason for this phenomenon, something unusual for a Markov decision chain, and an example, are given in this section.

The recurrence of values for some states before there is recurrence of values for all states, i.e. before value convergence, is connected to the persistence of hypermyopic decisions for some of the states. That

is, although policy improvements indicate that the initial, hypermyopic policy is nonoptimal, some of the decisions comprising that policy, i.e. decisions not to relocate the new facility, may persist into later iterations. For the following development it is useful to introduce the partitioning of the state space $S_1,S_2,\ldots,S_N$ where $S_i = \{(i,1),(i,2),\ldots,(i,N)\}$. The importance of the partitioning is that if the system is in state (i,j) at time t, then it is certain to be in some state belonging to $S_{k_{ij}}$ at time $t+1$.

When hypermyopic decisions are in use for all the states associated with a particular facility location the resulting Markov chain possesses a closed* class of states. For example if $k_{2j}=2$ for $j=1,\ldots,N$, at some iteration then the process will remain indefinitely in the class of states S_2 if it ever enters this class. In such cases, regardless of how the other decisions have been updated after the initial VDO, the values obtained for the closed class at the initial VDO are correct for later VDO 's. Indeed, if $k_{2j}=2$ for $j=1,\ldots,N$ is optimal, then the initial values of $v_{21},v_{22},\ldots,v_{2N}$ are optimal. This is true because

---

* The terminology used here is as follows: In reference to the Markov chain with transition matrix T resulting from a pure policy K,

- a *closed* class of states has the property that once it is entered it cannot be exited;
- an *ergodic* class of states is a minimal closed class, i.e. it contains no closed subclasses; an *absorbing* state is a singleton ergodic class;
- a *communicating* class of states has the property that every member of the class can be reached with positive probability in a finite number of steps from any other member of the class; a class C is communicating if and only if there exists a sequence $i_1,i_2,\ldots,i_u$ including all and only members of C such that the transition probability from state i_k to i_{k+1}, $k=1,\ldots,u-1$ and the transition probability from state i_u to i_1 are all positive. A class is ergodic if and only if it is closed and communicating.

the states in the closed class must satisfy

$$\hat{V}_2 = \hat{W}_2 + P\hat{V}_2$$

which is solved at the outset (3.12). Even when these states interact algebraically with other states outside the closed class, the values for the states in the class are independent of values outside the class. For example, if $k_{2j}=2$ for all j and $k_{47}=2$, then the v_{2j}'s appear on the right-hand side of the v_{47} equation; but, since state (4,7) can never by entered from state (2,j), there is no dependence of v_{2j} on v_{47}.

Not only should the values for states (2,j) be known prior to VDO calculations for as long as the class S_2 remains closed, but also the value for any state (i,j) having $k_{ij}=2$ should be known prior to the VDO. This follows because $k_{ij}=2$ guarantees that the system, if operated under the current policy and started in state (i,j), is certain to be absorbed by the closed class at the next transition, after which the expected long-term discounted costs are known. Moreover, when all the values v_{ij}, $j=1,\ldots,N$ for some i are known then so are the values known for any state (h,ℓ) such that $k_{h\ell}=i$, and so forth. This feature is not common to Markov decision chains. Even when the optimal decision for some state is used constantly throughout the solution of a more common Markov decision problem, the value associated with that state prior to termination is nonoptimal because this value depends on the current (nonoptimal) values of the other states.

This condition of partial knowledge of the state values simplifies

the value determination operation, particularly for an iterative numerical technique. Those v_{ij} 's which are known are omitted from the updating procedure. (Recurrence of values is also a reason for omitting VDO 's.)

Table 1 shows the sequence of policies used and values generated for a 16-state problem under Howard's policy iteration method. The hypermyopic policy was assumed initially. The figures in the policy improvement row (PIR) indicate the index and the value of

$$\min_k \left\{ \alpha f_{ik} + w_{jk} + \beta \sum_{\ell=1}^{N} p_{i\ell} v_{k\ell} \right\} \tag{3.19}$$

for state (i,j) , using values from the last VDO for the $v_{k\ell}$'s ; i.e. there is no feedback in the policy improvement routine. The data assumed were

$$P = \begin{pmatrix} .2 & .7 & 0 & .1 \\ .05 & .15 & .1 & .7 \\ .05 & .2 & .6 & .15 \\ .9 & .1 & 0 & 0 \end{pmatrix} \qquad F = \begin{pmatrix} 0 & 6 & 3 & 8 \\ 5 & 0 & 7 & 3 \\ 3 & 7 & 0 & 6 \\ 9 & 4 & 6 & 0 \end{pmatrix}$$

$$\alpha = 1.3 \qquad \beta = 0.9$$

All the VDO values which are marked with an asterisk in Table 1 correspond to v_{ij} 's which are known prior to the actual solution of the VDO equations. In this case all such occurences are due to the persistence of hypermyopic decisions in class S_2 . Further inspection of Table 1 indicates that some of the other v_{ij} 's are obtainable by more

Table 1. Solution History for 16-State Example Problem

STATE:		11	12	13	14	21	22	23	24	31	32	33	34	41	42	43	44	
Initial Policy		1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4	
VDO		44.56	44.25	42.75	40.68	28.47	30.67	34.34	30.32	50.02	48.52	43.38	48.28	46.26	45.76	47.67	50.09	
PIR	Policy	2	2	2	2	2	2	2	2	2	2	3	2	2	2	2	2	
	Value	36.27	38.47	42.14	38.13	28.47	30.67	34.34	30.32	37.57	39.77	43.38	39.42	33.67	35.87	39.54	35.52	
VDO		36.27*	38.47*	42.14*	38.13*	28.47*	30.67*	34.34*	30.32*	37.57*	39.77*	36.13	39.42*	33.67*	35.87*	39.54*	35.52*	
PIR	Policy	2	2	3	1	2	2	2	2	2	2	3	1	2	4	4	2	
	Value	36.27	38.47	40.03	33.44	28.47	30.67	34.34	30.32	37.57	39.77	36.13	37.43	33.67	33.95	38.97	35.52	
VDO		36.27*	38.47*	39.42	33.44*	28.47*	30.67*	34.34*	30.32*	37.57*	39.77*	35.52	37.43*	33.67*	33.42	37.34	35.52*	
PIR	Policy	2	1	1	1	2	2	2	2	2	3	3	1	2	4	4	2	OPTIMAL
	Value	36.27	38.24	38.56	33.44	28.47	30.67	34.34	30.32	37.57	39.18	35.52	37.35	33.67	33.42	37.34	35.52	
VDO		36.27*	37.96	37.33	33.40	28.47*	30.67*	34.34*	30.32*	37.57*	39.03	35.22	37.30	33.67*	33.42*	37.34*	35.52*	

direct means than the iteration scheme (3.17) which was used. For example in the second VDO v_{33} is the only unknown value for class $S_{k_{33}} = S_3$, so it may be obtained by the single computation

$$v_{33} = \frac{r_{33} + \beta(p_{31}v_{31} + p_{32}v_{32} + p_{34}v_{34})}{1 - \beta p_{33}} \tag{3.20}$$

since all the terms on the right are known. The third VDO can be executed nearly as easily. Since $p_{43} = p_{44} = 0$, the value of v_{i4} depends only on $v(k_{i4},1)$ and $v(k_{i4},2)$ and not on $v(k_{i4},3)$ or $v(k_{i4},4)$, for all i. In particular $k_{14} = k_{34} = 1$ so v_{14} and v_{34}, which depend only on the known quantities v_{11} and v_{12}, are immediately obtainable. Then, after v_{34} is calculated, v_{33} is the only unknown value in class $S_{k_{33}} = S_3$, so it can be computed again by equation (3.20). This step completes calculation of $\hat{V}_3$ whereby v_{13} is obtainable since $k_{13} = 3$. Finally, the only two remaining unknowns v_{42} and v_{43} require the solution of the system of two equations:

$$\begin{aligned}(1-\beta p_{22})v_{42} - \beta p_{23}v_{43} &= r_{42} + \beta p_{21}v_{41} + \beta p_{24}v_{44} \\ -\beta p_{22}v_{42} + (1-\beta p_{23})v_{43} &= r_{43} + \beta p_{21}v_{41} + \beta p_{24}v_{44}\end{aligned} \tag{3.21}$$

The ability to obtain some of the values by solving a reduced system occurs often in general. This is related to the existence of closed classes of states such as when the hypermyopic policy is used and the VDO system decomposes to N subsystems. In the case of v_{42} and v_{43} in the third VDO of the preceeding example the set $S_2 \cup S_4$ is closed whereby the values $\hat{V}_2$ and $\hat{V}_4$ are independent of $\hat{V}_1$ and $\hat{V}_3$.

3.2.3 The State Classification Algorithm (Natural Decomposition).

The observations of the preceding section support the general notion that state classification reduces the computational burden of value determination on nonergodic chains.

Let the rows and columns of T be permuted so that the resulting matrix is

$$\tilde{T} = \begin{array}{c} \\ E_1 \\ E_2 \\ \\ E_r \\ T_1 \\ T_2 \\ \\ \\ T_s \end{array} \begin{array}{c} \begin{array}{ccccccccc} E_1 & E_2 & & E_r & T_1 & T_2 & & & T_s \end{array} \\ \begin{bmatrix} A_1 & & & & & & & & \\ & A_2 & & & & & & & \\ & & \ddots & & & & & & \\ & & & A_r & & & & & \\ P_{11} & P_{12} & \cdots & P_{1r} & U_1 & & & & \\ P_{21} & P_{22} & & P_{2r} & Q_{21} & U_2 & & & \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & & \\ P_{s1} & P_{s2} & \cdots & P_{sr} & Q_{s1} & Q_{s2} & \cdots & U_s \end{bmatrix} \end{array} , \quad (3.22)$$

where the states are partitioned into closed classes E_i , $i=1,\ldots,r$, and transient classes* T_j , $j=1,\ldots,s$. Let V_{E_i} , V_{T_j} be the corresponding vectors of state values and let

$$\tilde{V} = (V_{E_1} \quad \cdots \quad V_{E_r} \quad V_{T_1} \quad \cdots \quad V_{T_s})^T .$$

Assume a similar array $\tilde{R}$ of the immediate costs. The VDO equation (3.2) with this reordering of the states,

---

* A transient class is a set of transient states.

$$\tilde{V} = \tilde{R} + \beta\tilde{T}\tilde{V} \quad , \tag{3.23}$$

is a block-triangular system

$$V_{E_i} = \tilde{R}_{E_i} + \beta A_i V_{E_i} \quad , \; i=1,\ldots,r \tag{3.24}$$

$$V_{T_1} = \tilde{R}_{T_1} + \beta \sum_{i=1}^{r} p_{1i} V_{E_i} + \beta U_1 V_{T_1} \tag{3.25}$$

$$V_{T_j} = \tilde{R}_{T_j} + \beta \sum_{i=1}^{r} P_{ji} V_{E_i} + \beta \sum_{k=1}^{j-1} Q_{jk} V_{T_k} + \beta U_j V_{T_j} \quad , \; j=2,\ldots,s \tag{3.26}$$

requiring for direct solution, at most, the inversion of $I-\beta A_i$, $i=1,\ldots,r$ and $I-\beta U_j$, $j=1,\ldots,s$; which is of course simpler than inverting $I-\beta T$.

The block-diagonal VDO corresponding to the hypermyopic policy is a special case of the above decomposition with $r=N$, $s=0$, $E_i=S_i$, and $A_i=P$. The sets S_i are always closed classes under the hypermyopic policy, and they are ergodic classes if and only if P is an ergodic matrix, i.e., if and only if the one-dimensional Markov chain describing existing-facility locations is ergodic.

The next question to address is, then, how to classify the states. Several forms of the $\tilde{T}$ matrix may be possible depending on how much effort is expended in making the number of separate blocks as large as possible which then makes the matrices to be inverted as small as possible. The form of the complete transition matrix which affords the largest possible number of blocks is the *canonical* form, defined here as follows: The matrix $\tilde{T}$ given by (3.22) is *canonical* if:

(i) the classes E_i , i=1,...,r are ergodic;

(ii) the classes T_j , j=1,...,s are communicating; and

(iii) if a state belonging to T_j can be reached from a state belonging to T_k , then $j \leq k$.

The classes $E_1,\ldots E_r,T_1,\ldots T_s$ of the canonical form will together be called the *canonical classes* $C_1,\ldots C_q$, respectively. The important property possessed by a canonical class C_i is that the state values associated with its members depend on each other and, if they depend on values of states in some other class C_j , then $j \leq i$. An algorithm for identifying the canonical classes is now presented.

This algorithm applies to the VDO for any Markov decision problem when it is suspected that the Markov chain resulting from the currently assumed policy possesses several distinct classes when put into its canonical form. The algorithm serves no purpose when the chain is ergodic, i.e. when only one canonical class exists. Experience indicates that the location problem falls into the class of problems to which the algorithm applies. At this point, the states are indexed with a single subscript for the purpose of clearly presenting the algorithm without clouding its applicability outside the location problem context. The algorithm bears some resemblence to the state classification algorithm of Fox and Landi [14] except that in Fox and Landi's approach the transient states are all lumped into one class.

Let p_{ij} be the transition probability from state i to state j . Let r_i be the immediate cost in state i under the current policy. The equations to be solved are,

$$v_i = r_i + \sum_{j \in S} p_{ij} v_j \quad , \; i \in S \; .$$

M is the set of "undeleted states," initially $M=S$, the state space. A Boolean matrix B_M is maintained with elements

$$b_{ij} = \begin{cases} 1 & \text{if } i \neq j \text{ and } p_{ij} > 0 \\ 0 & \text{otherwise} \end{cases} \quad , \; i,j \in M \; . \tag{3.27}$$

When a row of B_M is all zero it is called a *null row*. The state (or states) associated with a null row can have its value (or their values) determined; initially, the null rows correspond to absorbing states i, for which $v_i = \frac{r_i}{1-\beta}$. When a state is evaluated it is deleted from M and B_M is reduced in size. This process is repeated until no null rows exist. When that happens a search for a set of communicating states in M is conducted by generating a chain of states in M such that each state can be reached from its predecessor with positive probability in one transition. In extending the chain some state will eventually repeat (because all states in M lead to some other state in M when no null rows exist, and M is finite); the cycle comprises a communicating class which is collapsed into a single equivalent state. Then B_M is reduced in size again and the search for a null row is resumed. A null row corresponds to a canonical class, because the states (state) associated with the null row comprise(s) an ergodic class or the states (state) of the null row lead(s) only to states, deleted from M, whose values have already been determined. Steps of the algorithm are:

Step 0: Set $M \leftarrow S$, $C_i \leftarrow \{i\}$ for all $i \in S$. Set b_{ij} according to (3.27). Set $L \leftarrow \phi$. (L is the set of evaluated states.)

Step 1: Look for a null row in B_M . If there is none, go to Step 4.

Step 2: Say row k of B_M is the null row.

a) Set $M \leftarrow M - C_k$

b) For all $i \in C_k$ set $r_i' \leftarrow r_i + \beta \sum_{j \in L} p_{ij} v_i$.

c) Solve $v_i = r_i' + \beta \sum_{j \in C_k} p_{ij} v_j$, $i \in C_k$.

d) Set $L \leftarrow L \cup C_k$.

(C_k is a canonical set and its state values are computed by solving the smallest possible system.)

Step 3: If $L = S$, terminate; otherwise go to Step 1.

Step 4:

a) Set $\ell \leftarrow 1$. Set i_ℓ equal to any state in M .

b) Set $i_{\ell+1}$ equal to any state in M such that $b(i_\ell, i_{\ell+1}) = 1$.

c) If $i_{\ell+1} = i_k$ for some $k \leq \ell$ go to Step 4-d; otherwise set $\ell \leftarrow \ell+1$ and go to Step 4-b.

d) (The cycle $i_k, i_{k+1}, \ldots, i_\ell, i_k$ is identified.)

(i) Replace row i_k with the union of rows $i_k, i_{k+1}, \ldots, i_\ell$; i.e. if $b(i_h, j) = 1$ for some $h = k, k+1, \ldots, \ell$, $j \in M$, then set $b(i_k, j) \leftarrow 1$.

(ii) Replace column i_k with the union of columns $i_k, \ldots, i_\ell$.

(iii) Reset $b_{i_k,i_k} \leftarrow 0$.

(iv) Set $C_{i_k} \leftarrow C_{i_k} \cup C_{i_{k+1}} \ldots \cup C_{i_\ell}$.

(v) Set $M \leftarrow M - (C_{i_{k+1}} \cup \ldots \cup C_{i_\ell})$.

e) If row i_k is null in M go to Step 2; otherwise set $\ell \leftarrow k$ and go to Step 4-b.

The method of solution for Step 2-c is not specified; any method, direct or iterative, will do. Most likely, the cardinality of C_k determines which method to use. In particular, successive overrelaxation is still feasible; in fact, the subsystem of Step 2-c is at least as diagonally dominant as the original system.* The importance of the algorithm is to decompose the VDO , to determine which states need to be evaluated simultaneously and in what order the evaluations should be done.

Decomposition ideas have been applied previously to Markov decision processes. [8,17,29,59] Decomposing the value determination

---

* The diagonal elements of both the original system, $v_i = r_i + \beta \sum_{j \in S} p_{ij} v_j$, $i \in S$, and the subsystem $v_i = r_i' + \beta \sum_{j \in C_k} p_{ij} v_j$, $i \in C_k$, are $\gamma_i = 1 - \beta p_{ii}$. The off-diagonal elements for row i of the original system sum to $\gamma_i' = -\beta(1-p_{ii}) \leq 0$; the off-diagonal for row i of the subsystem sums to $\gamma_i'' = -\beta(1-p_{ii} - \sum_{j \in S-C_k} p_{ij}) \leq 0$. The subsystem is at least as diagonally dominant as the original system since $|\gamma_i| - |\gamma_i''| = \gamma_i + \gamma_i'' = 1 - \beta + \beta \sum_{j \in S-C_k} p_{ij} \geq 1 - \beta = |\gamma_i| - |\gamma_i'|$.

operation using state classification is implicit in Fox and Landi's paper [14] ; however, the state classification method introduced here results in a finer partitioning of the state space. We shall call the decomposition based on canonical classes *natural decomposition.*

We next present an example of natural decomposition. Consider the single-facility dynamic probabilistic location problem with three possible locations, with

$$P = \begin{pmatrix} .801 & .198 & .001 \\ .076 & .451 & .473 \\ 0 & 0 & 1 \end{pmatrix} , \quad F = \begin{pmatrix} 0 & 7.942 & 8.099 \\ 7.942 & 0 & 2.167 \\ 8.099 & 2.167 & 0 \end{pmatrix}$$

$$\alpha = 1.3 , \qquad \beta = .9 ;$$

and evaluate the policy

$$K = \begin{pmatrix} 1 & 3 & 3 \\ 1 & 2 & 3 \\ 1 & 3 & 3 \end{pmatrix} .$$

The expected immediate rewards

$$r_{ij} = \alpha f(i,k_{ij}) + \sum_{\ell=1}^{3} p_{j\ell} \, f(k_{ij},\ell)$$

are

$$R = \begin{pmatrix} 1.5806 & 12.1215 & 10.5287 \\ 11.9052 & 1.6286 & 2.8171 \\ 12.1093 & 1.5928 & 0 \end{pmatrix} .$$

The complete transition matrix is

$$T = \begin{array}{c|ccc|ccc|ccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline 1 & .801 & .198 & .001 & & & & & & \\ 2 & & & & & & & .076 & .451 & .473 \\ 3 & & & & & & & & & 1.0 \\ \hline 4 & .801 & .198 & .001 & & & & & & \\ 5 & & & & .076 & .451 & .473 & & & \\ 6 & & & & & & & & & 1.0 \\ \hline 7 & .801 & .198 & .001 & & & & & & \\ 8 & & & & & & & .076 & .451 & .473 \\ 9 & & & & & & & & & 1.0 \end{array} .$$

Single subscripts will be used to index states in the example. Initially, $M = S$ and

$B_M =$

	1	2	3	4	5	6	7	8	9
1		1	1						
2							1	1	1
3									1
4	1	1	1						
5				1		1			
6									1
7	1	1	1						
8							1		1
9									

Row 9, referring to the absorbing state (3,3) is a null row. Its value is $v_9 = r_9/(1-\beta) = 0$. With state 9 deleted from M , row 3 and row 6 become null rows. The values $v_3 = r_3 + \beta v_9 = 10.5287$ and $v_6 = r_6 + \beta v_9 = 2.8171$ are obtained for the singleton canonical classes $C_3 = \{3\}$ and $C_6 = \{6\}$. Deleting 3 and 6 from M ,

$B_M =$

	1	2	4	5	7	8
1		1				
2					1	1
4	1	1				
5			1			
7	1	1				
8					1	

and there are no null rows. We then enter the cycle step, Step 4, and find that states 1, 2 and 7 communicate, that is $b_{12} = b_{27} = b_{71} = 1$.

By Step 4-d we collapse the rows and columns for the communicating class into a single row and a single column (which have label 1), whence

$$B_M = \begin{array}{c} \\ 1 \\ 4 \\ 5 \\ 8 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 4 & 5 & 8 \end{array} \\ \begin{pmatrix} & & & 1 \\ 1 & & & \\ & 1 & & \\ 1 & & & \end{pmatrix} \end{array} .$$

Note $b_{18} = 1$ because previously $b_{28} = 1$ (Step 4-d(i)); $b_{81} = 1$ because $b_{87} = 1$ (Step 4-d(ii)).

B_M still does not have a null row so we resume the search for a cycle beginning with state 1, the state that represents the previously identified cycle (Step 4-e). The cycle $1 \to 8 \to 1$ is immediately found; state 8 is augmented to the communicating class $C_1 = \{1,2,7,8\}$ and deleted from M. At this point

$$B_M = \begin{array}{c} \\ 1 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 4 & 5 \end{array} \\ \begin{pmatrix} & & \\ 1 & & \\ & 1 & \end{pmatrix} \end{array}$$

and row 1 is null. Thus $\{1,2,7,8\}$ is identified as a canonical class which is now ready to be evaluated. The right-hand side of the four by four system, which needs to be solved to evaluate this class, is updated in Step 2-b. States 1 and 7 lead to the previously evaluated state 3 so the terms $\beta p_{13} v_3$ and $\beta p_{73} v_3$ appearing in the respective value

equations are added to r_3 and r_7. The system to be solved in Step 2-c is then

$$v_1 = 1.5806 + .9(.001)(10.5287) + .9(.801)v_1 + .9(.198)v_2$$

$$v_2 = 12.1215 + .9(.076)v_7 + .9(.451)v_8$$

$$v_7 = 12.1093 + .9(.001)(10.5287) + .9(.801)v_1 + .9(.198)v_2$$

$$v_8 = 1.5928 + .9(.076)v_7 + .9(.451)v_8$$

which results in

$$\begin{pmatrix} v_1 \\ v_2 \\ v_7 \\ v_8 \end{pmatrix} = \begin{pmatrix} 16.088 \\ 16.2742 \\ 26.6167 \\ 5.7455 \end{pmatrix} .$$

At this point only states 4 and 5 are remaining in M and

$$B_M = \begin{array}{c} \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cc} 4 & 5 \end{array} \\ \begin{pmatrix} & \\ 1 & \end{pmatrix} \end{array} .$$

Row 4 is null, indicating state 4 is ready to be evaluated:

$$v_4 = 11.9052 + .9(.801v_1 + .198v_2 + .001v_3) = 26.4126 .$$

Finally, state 5 whose value depends on v_4 and v_6 is computed from

$$(1-.9(.451))v_5 = 1.6286 + .9(.076)(26.4126) + .9(.473)(2.8171)$$

whence $v_5 = 7.8008$.

It was not actually necessary to rearrange the complete transition matrix into canonical form when implementing natural decomposition. The canonical form of T can be obtained however by simply noting the order in which states were evaluated: 9,3,6,1,2,7,8,4,5 . Thus,

$\tilde{T}$ =

	9	3	6	1	2	7	8	4	5
9	1								
3	1								
6	1								
1		.001		.801	.198				
2	.473					.076	.451		
7		.001		.801	.198				
8	.473					.076	.451		
4		.001		.801	.198				
5			.473					.076	.451

is the canonical form of T . The computations performed in the algorithm are equivalent to inverting a (properly-sized) identity matrix minus β times the matrix on each block of the highlighted block diagonal shown above.

Finally, if the Fox-Landi [14] classification algorithm, which does not differentiate between transient states, had been used to decom-

pose the value determination, then all states except the absorbing state 9 would have been grouped in a single block.

3.3 Partial Feedback Methods of Policy Improvement

This section presents alternative methods for policy improvement with respect to the single-facility location problem. Three policy improvement operators are considered and their computational aspects compared.

When Hastings [18,19] introduced the feedback concept in the Markov decision chain context, he said it "is generally to be preferred although it can not always be guaranteed to be more efficient in every case." [19] Of course, it is possible to determine which of two policy improvement routines requires less computational effort per application; the uncertainty arises in determining which method will need to be applied a smaller number of times in solving a given problem. The experience of Contreras [8] concurred with Hastings' view that feedback (Gauss-Seidel) policy improvements are preferred.

The non-feedback (Jacobi) policy improvement operator, characteristic of Howard's policy improvement algorithm, was applied to the example location problem in Section 3.2.2. This operator transforms V to ΠV where

$$(\Pi V)_{ij} = \underset{k=1,\ldots,N}{\text{minimum}} \left\{ \alpha f_{ik} + w_{jk} + \beta \sum_{\ell=1}^{N} p_{j\ell} v_{k\ell} \right\} . \qquad (3.28)$$

The full feedback policy improvement operator transforms V to $\tilde{\Pi} V$ where

$$(\tilde{\Pi}V)_{ij} = \min \left\{ \begin{array}{l} \min\limits_{k=1,\ldots,i-1} \left\{ \alpha f_{ik} + w_{jk} + \beta \sum\limits_{\ell=1}^{N} p_{j\ell} (\tilde{\Pi}V)_{k\ell} \right\} , \\ \alpha f_{ii} + w_{ji} + \beta \sum\limits_{\ell=1}^{j-1} p_{j\ell} (\tilde{\Pi}V)_{i\ell} + \beta \sum\limits_{\ell=j}^{N} p_{j\ell} v_{i\ell} , \\ \min\limits_{k=i+1,\ldots,N} \left\{ \alpha f_{ik} + w_{jk} + \beta \sum\limits_{\ell=j}^{N} p_{j\ell} v_{k\ell} \right\} . \end{array} \right. \tag{3.29}$$

In this instance the nonfeedback routine can be executed in about $\frac{1}{N} \times 100$ percent of the time required to execute the full feedback routine. The considerable time difference is due to the fact that the summation term representing discounted future costs in (3.28) is independent of i, whereas the summation terms in (3.29) must be computed individually for each i,j,k combination.

The section of FORTRAN code appearing in Figure 9(a) computes ΠV. The number of arithmetic operations required is on the order of N^3 as can be seen from the level of nesting of loops. A temporary storage array $S = \{s_{jk}\}$ is first filled with the values

$$s_{jk} = w_{jk} + \beta \sum_{\ell=1}^{N} p_{jk} v_{k\ell} \quad , \; j,k=1,\ldots,N \; ,$$

and then the minimum of $\{\alpha d_{ik} + s_{jk} \mid k=1,\ldots,N\}$ is selected and stored in v_{ij} for each i,j. The minimizing index is stored in k_{ij} and if it is different from the previous value of k_{ij}, a flag indicating lack of policy convergence is set. (At completion of the codes of Figure 9 ICNVRG = 1 if and only if policy convergence occurs.)

The code in Figure 9(c) computes $\tilde{\Pi}V$. It does not require a working array but the number of operations it performs is on the order

of N^4 . The four levels of nested loops are needed to compute the discounted future cost expressions using the most recently obtained state values. Assuming storage space is available for the working array S , full feedback is not worthwhile unless its absense causes an N-fold increase in the number of policy improvements required to solve a problem. Among the examples we have considered this has never been the case; as shown in Section 4, fewer than N iterations of policy improvement are usually necessary to finish a problem.

We next present a partial feedback policy improvement routine whose execution time is the same as that of the nonfeedback routine. It only requires an N×1 working array, as compared to the N×N working array used in the nonfeedback routine. In addition to its storage advantage over nonfeedback, partial feedback has demonstrated in test problems the macro convergence advantage (i.e. reducing the number of policy improvements required) attributed to full feedback. We feel it is the method of choice and give computational experience to support this belief in Section 4.

Partial feedback, as indicated in Chapter II, involves the use of some but not all of the most recently obtained information when revising values in the policy improvement routine. In particular, for the single-facility location problem a partial feedback policy improvement operator is $\overline{\Pi}^E V$ where

$$\left(\overset{E}{\Pi V}\right)_{ij} = \min \left\{ \begin{array}{l} \min\limits_{k=1,\ldots,i} \left\{ \alpha f_{ik} + w_{jk} + \beta \sum\limits_{\ell=1}^{N} p_{j\ell} \left(\overset{E}{\Pi V}\right)_{k\ell} \right\} \quad , \\ \min\limits_{k=i+1,\ldots,N} \left\{ \alpha f_{ik} + w_{jk} + \beta \sum\limits_{\ell=1}^{N} p_{j\ell} v_{k\ell} \right\} \end{array} \right. \tag{3.30}$$

The aforementioned advantages of $\overset{E}{\Pi}$ are realized by considering the state values in the order $v_{11}, v_{21}, \ldots, v_{N1}, \ldots, v_{1N}, \ldots, v_{NN}$, instead of in the usual order. For each j the scratch array $S = \{s_k\}$ is filled with the values.

$$s_k = w_{jk} + \beta \sum_{\ell=1}^{N} p_{j\ell} v_{k\ell} \quad , \; k=1,\ldots,N \; ;$$

the minimum of $\{\alpha d_{ik} + s_k \mid k=1,\ldots,N\}$ is then selected and stored in v_{ij} for states $(1,j),(2,j),\ldots,(N,j)$, after which j is increased by one and the process repeats. The code in Figure 9(b) implements this procedure.

3.4 Action Elimination

MacQueen's test for elimination of suboptimal actions (Section 6.2 of Chapter II) is next developed for the single-facility location problem. As noted in Chapter II, the state-change formulation offers computational advantages in connection with this test. The MacQueen test for the location problem is: *Moving the new facility to location* h *is nonoptimal in state* (i,j) *if*

$$\alpha f_{ih} + w_{jh} + \beta \sum_{\ell=1}^{N} p_{j\ell} v_{h\ell} > \min_{k} \left\{ \alpha f_{ik} + w_{jk} + \beta \sum_{\ell=1}^{N} p_{j\ell} \underline{v}_{k\ell} \right\} \quad ,$$

```
Read V,K,P,F,W,ALPHA,BETA

   ICNVRG=1
   DØ 10 J=1,N
   DØ 10 M=1,N
   S(J,M)=0.
   DØ 20 L=1,N
20 S(J,M)=S(J,K)+P(J,L)*V(M,L)
10 S(J,M)=BETA*S(J,M)+W(J,M)

   DØ 30 I=1,N
   DØ 30 J=1,N
   VNEW=10.E10
   DØ 40 M=1,N
   VTRIAL=ALPHA*F(I,M)+S(J,M)
   IF(VNEW.LE.VTRIAL) GØ TØ 40
   VNEW=VTRIAL
   KNEW=M
40 CONTINUE
   IF(KNEW.EQ.K(I,J)) GØ TØ 30
   K(I,J)=KNEW
   ICNVRG=0
30 V(I,J)=VNEW

Return V,K,ICNVRG
```

a) Nonfeedback

```
Read V,K,P,F,W,ALPHA,BETA

   ICNVRG=1
   DØ 30 J=1,N
   DØ 10 M=1,N
   S(M)=0.
   DØ 20 L=1,N
20 S(M)=S(M)+P(J,L)*V(M,L)
10 S(M)=BETA*S(M)+W(J,M)

   DØ 30 I=1,N

   VNEW=10.E10
   DØ 40 M=1,N
   VTRIAL=ALPHA*F(I,M)+S(M)
   IF(VNEW.LE.VTRIAL) GØ TØ 40
   VNEW=VTRIAL
   KNEW=M
40 CONTINUE
   IF(KNEW.EQ.K(I,J)) GØ TØ 30
   K(I,J)=KNEW
   ICNVRG=0
30 V(I,J)=VNEW

Return V,K,ICNVRG
```

b) Partial Feedback

```
Read V,K,P,F,W,ALPHA,BETA

   ICNVRG=1
   DØ 30 I=1,N
   DØ 30 J=1,N
   VNEW=10.E10
   DØ 40 M=1,N
   VTRIAL=0.
   DØ 50 L=1,N
50 VTRIAL=VTRIAL+P(J,L)*V(M,L)
   VTRIAL=ALPHA*F(I,M)+W(J,M)+BETA*VTRIAL
   IF(VNEW.LE.VTRIAL) GØ TØ 40
   VNEW=VTRIAL
   KNEW=M
40 CONTINUE
   IF(KNEW.EQ.K(I,J)) GØ TØ 30
   K(I,J)=KNEW
   ICNVRG=0
30 V(I,J)=VNEW

Return V,K,ICNVRG
```

c) Full Feedback

Figure 9. FORTRAN Codes for Policy Improvement Operators

where $\underline{V} \leq V^* \leq \overline{V}$. The computational advantage is related to the lack of dependence on i in the summation terms. Unfortunately, the additional storage required to keep track of the nonoptimal actions for each state (i,j) , as well as the bounds, somewhat restricts the applicability of the test.

4. Computational Experience

Special characteristics of the single-facility dynamic probabilistic location problem were described and exploited in the development of components of solution techniques in Section 3. In this section, the components are assembled into an algorithm whose performance on test problems is reported.

As noted in Chapter II, when designing an algorithm for Markov decision chains, there are many options available in terms of macro structure and micro structure. By macro structure we refer to the way in which components (e.g., initialization, policy improvement and policy evaluation operators) are assembled; by micro structure we mean choosing from among the possible versions for each component.

The purpose of this section is to demonstrate that, at least for all the test cases considered, the single-facility dynamic probabilistic location problem is tractable, in spite of what may at first seem to be an intractable state space. We do not purport to have subjected the large set of possible algorithms (nor even a known finite subset) to scientific comparison; the algorithm reported here is good but better ones may exist.

The macro structure we have chosen is the delayed evaluation

algorithm discussed in Chapter II. It uses Porteus's [35] suggestion to apply the policy improvement operator to an initial set of values as many times as necessary to achieve policy convergence; then, if value convergence fails to hold, a value determination and one policy improvement are performed. If policy convergence does not hold, the algorithm repeats. The micro structure is: use the hypermyopic policy initially and its associated values, use partial feedback for policy improvement, and use successive overrelaxation for value determination. The action elimination procedure, which may be viewed as re-initialization in terms of macro structure, is not used because of its storage disadvantages. The algorithm is flow-charted in Figure 10.

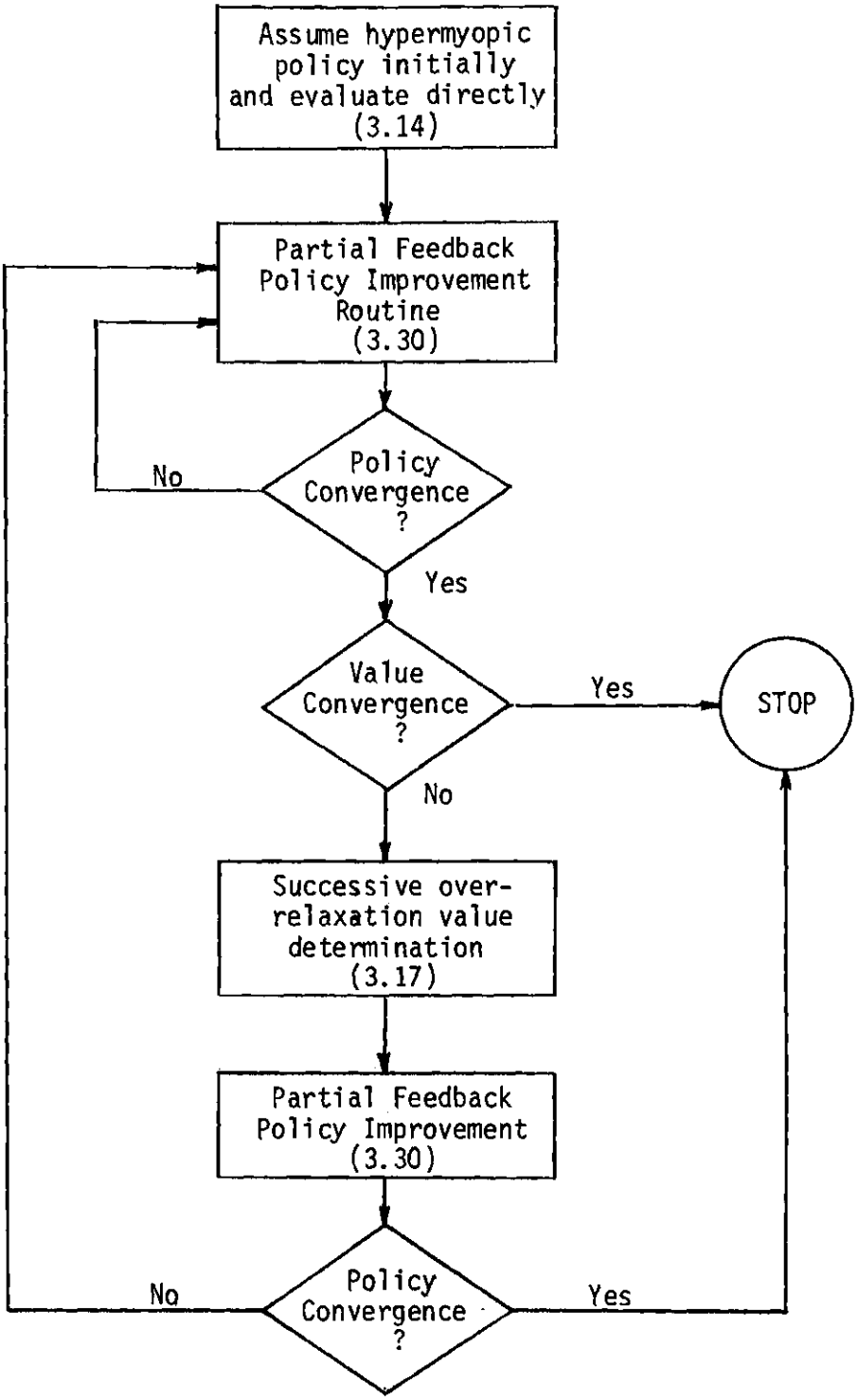


Figure 10.

Algorithm for Solving Single-Facility Problem

The performance of this algorithm on test problems is reported in Table 2. The method of problem generation is discussed in the Appendix. For four problem sizes, ranging in number of states from 16 to 2500, the results reported in the table are: the average and maximum number of policy improvements, the distribution of the number of value determinations and the average computation time (CPU seconds on a Univac 1108 computer) expended on one of these operations.

Table 2. Single-Facility Computational Experience

N	Policy Improvement				Value Determination*			
	Number of Problems	No. of Iterations		CPU Time per Iteration (sec)	No. of problems having:			CPU Time per Iteration (sec)
		Avg.	Max.		0 VDO's	1 VDO's	2 VDO's	
6 (36 states)	9	6.222	9	.006	0	5	4	.099
10 (100 states)	10	5.8	7	.024	2	7	1	.219
20 (400 states)	13	8.6	10	.163	4	9	0	.579
50 (2500 states)	1	11	11	2.711	1	0	0	--

* Excluding initial VDO on hypermyopic policy

The values of α and β were varied considerably, with β as high as .9999, but performance of the algorithm did not appear sensitive to these changes. The two most encouraging aspects of the computational results are that 1) the number of policy improvements required to solve the problem did not increase with problem size, thus insuring that the partial feedback PIR is preferred to the N-times-less-efficient full feedback PIR; and 2) no more than two value determinations were required after the efficient initial VDO (3.14) on the hypermyopic policy. For a large proportion of the problems the number of VDO 's after the initial VDO was 0 or 1. This would indicate that for many problems the policy given by the first policy convergence would be optimal. A heuristic procedure using 1) (3.14) to initialize the values, 2) partial feedback policy improvement for updating and 3) policy convergence as a termination rule, would then have a good chance of finding the optimum. The key to the success of this "successive approximation" procedure is the quality of the starting solution obtained from the hypermyopic policy.

CHAPTER IV

MULTIFACILITY DYNAMIC PROBABILISTIC LOCATION PROBLEMS

Chapter III demonstrates that the single-facility dynamic probabilistic location problem is tractable in spite of what at first might seem to be a prohibitively large state space. This chapter considers the question: How does the introduction of multiple existing facilities alter the modeling of the process and how does it affect our ability to compute solutions?

The first section to follow extends the work of Chapter III by modeling and solving the multifacility problem as a state-change Markov decision chain. Computational experience is reported for example problems with three existing facilities. The next two sections propose approximate procedures for larger problems.

1. Extension of the Single-Facility Approach

1.1 Modeling the Process

Let there by M existing facilities. As in Chapter III it is assumed that the locations of all facilities change over time and are confined to a finite set $N = \{1,\ldots,N\}$ of possible locations. The state of the process at time t, $t=0,1,2,\ldots$, is represented as an $(M+1)$-tuple, $(X_t,A_{1t},\ldots,A_{Mt})\in N^{M+1}$, where X_t, a decision variable, is the location of the new facility at time t, A_{mt} is the location of the m^{th} existing facility at time t, and N^{M+1} is the Cartesian product

space $\overset{M+1}{\underset{1}{\mathbf{X}}} N$. The changing locations of the existing facilities are described by independent Markov transition laws. Precisely, for $m=1,\ldots,M$, there is a known Markov transition matrix P_m whose elements are

$$p_m(j,\ell) = \Pr\{A_{m,t+1} = \ell \mid A_{mt} = j\} \quad , \; j,\ell \in N; \; t=0,1,\ldots ; \qquad (1.1)$$

and, by the independence assumption,

$$\Pr\{A_{1,t+1}=\ell_1,\ldots,A_{M,t+1}=\ell_M \mid A_{1t}=j_1,\ldots,A_{Mt}=j_M\} = \prod_{m=1}^{M} p_m(j_m,\ell_m) \; . \qquad (1.2)$$

An arbitrary realization of the M+1-dimensional state variable is written as $(i,j_1,j_2,\ldots,j_M)$ or as (i,J) where $J \in N^M$. The next realization is $(k,\ell_1,\ell_2,\ldots,\ell_M)$ or (k,L) where $L \in N^M$. The transition probabilities of equation (1.2) are written concisely as $p(J,L)$,

$$p(J,L) = p\big((j_1,\ldots,j_M),(\ell_1,\ldots,\ell_M)\big) = \prod_{1}^{M} p_m(j_m,\ell_m) \; . \qquad (1.3)$$

Note, the function $p(J,L)$ in (1.3), which maps $N^M \times N^M$ into $[0,1]$, serves the same purpose as $p(j,\ell)$ in the single-facility model, namely, to describe the (chance-determined) relocation of existing facilities in each time period.

Our presentation of the multifacility model assumes that costs are proportional to distance, a condition which is convenient for generating and processing data for test problems, but not essential for our methods to work. Relocating the new facility from location i to k has cost $\alpha_0 f_{ik}$. The service activity takes place in one of two ways.

Either the existing facilities go to the new facility for service in each time period; or the new facility visits each existing facility one at a time, always returning to the starting location between visits. The latter type of service is referred to as a *Weber tour* [58]. (The order in which existing facilities are visited on a Weber tour does not affect the total distance traveled, in contrast to a *traveling salesman tour* [58].) With either type of service--existing facilities visiting the new facility or the new facility taking a Weber tour--the cost of service is proportional to a weighted sum of the distances separating the new facility from all the existing facilities. Thus, if the new facility is at location i and the existing facilities have the single space M-tuple location $J=(j_1,\ldots,j_M)$ then the service cost is $\sum_{m=1}^{M} \alpha_m f(i,j_m)$.

The weighting factor α_m , $m=1,\ldots,M$, may have several interpretations. In addition to the interpretations common to location analysis [15], such as those referring to unit transportation costs or distribution volumes, there is another interesting interpretation of the weighting factor. Suppose in each time period that there is a probability α_m that the m^{th} existing facility will request service from the new facility and a probability $1-\alpha_m$ that it will not; then $\alpha_m f(i,j_m)$ is the expected distance traveled for the rendering of that uncertain service. The data requirements for the multifacility model are:

N , the number of possible locations

M , the number of existing facilities

P_m , $N\times N$ Markov transition matrix, $m=1,\ldots,M$

F , $N\times N$ distance matrix

α_0 , relocation cost factor

α_m , m=1,...,M , relative service cost factors

β , discount factor.

The order of events is:

1) The decision-maker observes $(X_{t-1},A_{1,t-1},\ldots,A_{M,t-1})$ and chooses X_t ;

2) The relocation cost $\alpha_0 f(X_{t-1},X_t)$ is incurred;

3) The chance locations $(A_{1t},\ldots,A_{Mt})$ are realized;

4) The service cost $\sum_{m=1}^{M} \alpha_m f(X_t,A_{mt})$ is incurred; and the process repeats.

The objective is to minimize the expected present worth of all costs,

$$\text{Minimize } E\left\{\sum_{t=1}^{\infty}\left[\alpha_0 f(X_{t-1},X_t) + \sum_{m=1}^{M}\alpha_m f(X_t,A_{mt})\right]\beta^{t-1}\right\} . \qquad (1.4)$$

In general the service cost is $\sum_{m=1}^{M} f_m'(X_t,A_{mt})$ where F_m' is the service cost matrix for existing facility m . We assume $F_m' = \alpha_m F$.

1.2 State-Change Formulation

Problem (1.4) is represented in this section as a state-change Markov decision chain whose optimal solution is characterized by a system of extremal equations. This development closely resembles the analagous results of Chapter III, and is, thus, reported in concise fashion.

The transition $(i,J) \to (k,L)$ in a typical time period is viewed as having two phases. The decision maker, upon observing (i,J) and deciding where to relocate the new facility, effects the state change $(i,J) \to (k,J)$. Then, "Nature" effects the chance transition $(k,J) \to (k,L)$ by changing the locations of the existing facilities. The probability associated with the first state change is $d(i,J,k)$; the prob-

ability of the second is $p(J,L)$ given by (1.3). The state-change probabilities are decision variables,

$$d(i,J,k) = 1 \text{ , if the decision maker chooses to move the new facility to location } k \text{ whenever state } (i,J) \text{ is observed;}$$

$$= 0 \text{ , otherwise.}$$

Note that pure, stationary policies are assumed; the arguments of previous chapters apply to justify this assumption. The alternate notation for a pure policy is $K = \{k(i,J)\}$ where $k(i,J) \in N$ and $d(i,J,k(i,J)) = 1$, for all $(i,J) \in N^{M+1}$.

The costs incurred in a time period are the state-change (relocation) cost and the state-visit (service) cost. The conditional expectation, $r(i,J)$, of the one period (immediate) costs, given state (i,J) at the start of the period, is next derived. $r(i,J)$ is the expected relocation cost plus the expected service costs for each existing facility. Let

$$w_m(j,k) = \sum_{\ell \in N} p_m(j,\ell) f_{k\ell} \quad , \; j,k \in N \; ; \; m=1,\ldots,M \; . \tag{1.6}$$

$w_m(j,k)$ is the expected unweighted service cost for existing facility m , and the matrix $W_m = \{w_m(j,k)\}_{j,k \in N}$ satisfies

$$W_m = P_m F^{\tau} \quad , \; m=1,\ldots,M \; . \tag{1.7}$$

The expected cost of service of all facilities, when the existing facilities are at locations $J = (j_1,\ldots,j_M)$ and the new facility has just

been moved to location k, is $w(J,k)$ where

$$w(J,k) = \sum_{m=1}^{M} \alpha_m w_m(j_m,k) \quad . \tag{1.8}$$

The cost of the state change $(i,J) \to (k,J)$ is $\alpha_0 f_{ik}$ and since this change occurs with probability $d(i,J,k)$, the expected one period cost is

$$r(i,J) = \sum_{k \in N} d(i,J,k)\left[\alpha_0 f_{ik} + w(J,k)\right] \quad . \tag{1.9}$$

Given a pure state-change policy K, (1.9) reduces to

$$r(i,J) = \alpha_0 f\left[i,k(i,J)\right] + w\left[J,k(i,J)\right] \tag{1.10}$$

for all $(i,J) \in N^{M+1}$.

The value determination equations for the multi-facility problem have the form

$$v(i,J) = r(i,J) + \beta \sum_{(k,L) \in N^{M+1}} t\left[(i,J),(k,L)\right] v(k,L) \tag{1.11}$$

where the complete transition probabilities are

$$t\left[(i,J),(k,L)\right] = \begin{cases} p(J,L) & \text{, if } k=k(i,J) \\ 0 & \text{, otherwise} \end{cases} \quad . \tag{1.12}$$

Substituting (1.10) and (1.12) in (1.11) gives for the value determination equations

$$v(i,J) = \alpha_0 f\big(i,k(i,J)\big) + w\big(J,k(i,J)\big) + \beta \sum_{L\epsilon N^M} p(J,L)v\big(k(i,J),L\big) \quad (1.13)$$

for all $(i,J)\epsilon N^{M+1}$. Equation (1.13) is identical to equation (2.16) of Chapter III except that M-tuple locations J and L are used in place of single-component locations j and ℓ.

The extremal equations for the multifacility dynamic probabilistic location problem, obtained from (1.13) by replacing the $k(i,J)$'s with the locations that minimize the state values, are

$$v(i,J) = \min_k \left\{ \alpha_0 f_{ik} + w(J,k) + \beta \sum_{L\epsilon N^M} p(J,L)v(k,L) \right\} \quad (1.14)$$

for all $(i,J)\epsilon N^{M+1}$.

1.3 Exact Solution Procedure

We next present an exact solution procedure for the state-change Markov decision chain model of the multifacility location problem. The procedure, extending the methods developed previously for the single-facility problem, has been coded for computer and applied to example problems with three existing facilities and as many as six possible locations (a total of $6^4 = 1296$ states).

As in the single facility case, we are fortunately not required to store the N^{M+1} by N^{M+1} complete transition matrix T. This is due to the special structure of T indicated by equation (1.12). Unfortunately we must choose between the Scylla of storing the N^M by N^M matrix of existing facility transition probabilities $P = \{p(J,L)\}$ and the Charybdis of having to reevaluate these quantities by (1.3) each time they are needed. The latter option, requiring storage of the MN^2

factors $p_m(j,\ell)$ instead of the N^{2M} probabilities $p(J,L)$, was taken, but at considerable cost in computation time due to the oft repeated multiplications in (1.3). Under this option the total storage required is a little more than $3N^{M+1}$ words with V, W and K constituting the bulk of it. (With $N = 6$ and $M = 3$, $N^{2M} = 46656$ and $3N^{M+1} = 3888$.) Storage can be cut further at additional expense in computation time; for example W could be reevaluated as needed instead of computed once and saved. Another storage economy device would be to place V and K in the same array A, say, where $a(i,J) = 10^8 \cdot k(i,J) + v(i,J)$. Encoding and decoding the A array would of course require extra computation which might be, with computers that carry about 8 significant digits (32 bits per word), prone to error.

The three main features of the exact procedure are, as usual for solving Markov decision chains, initialization, policy improvement and value determination. The macro structure into which these are assembled is the same macro structure which was applied to the single-facility problem and flow charted in Chapter III.

If the hypermyopic policy (zero relocation cost),

$$k(i,J) = i \quad , (i,J) \in N^{M+1} \quad , \tag{1.15}$$

is assumed initially, then the system of value determination equations (1.13) decomposes, in the same manner as the single-facility problem, to N separate systems

$$v(i,J) = w(J,i) + \beta \sum_{L \in N^M} p(J,L)v(i,L) \quad , J \in N^M \quad . \tag{1.16}$$

There is one subsystem (1.16) for each $i \varepsilon N$; each of these subsystems has the same coefficient matrix,

$$Q = I - \beta P$$

where $P = \{p(J,L)\}$. The matrix Q is of order N^M and, although Q is considerably easier to invert than the coefficient matrix $I - \beta T$ of order N^{M+1} in (1.13), we have chosen not to invert Q . (This is consistent with our choice not to store P .) Because inverting Q would be necessary for exploiting the commonality in structure of the subsystems (1.16), the hypermyopic policy is not as beneficial for finding initial values in the multifacility problem as it is in the single-facility problem.

The preferred value initialization procedure is the one given by (6.15) in Chapter II:

$$V^{(0)} = \Pi\left(\frac{1}{1-\beta}\Pi(0)\right) \quad . \tag{1.17}$$

The myopic values (smallest immediate costs) in the array $\Pi(0)$ are first determined and divided by $1-\beta$. The effect of dividing by $1-\beta$ is to raise the myopic values to a level more appropriate for the expected present worth of an infinite stream of cash flows. These values are then assumed for the first policy improvement which results in $V^{(0)}$.

The policy improvement routine of choice is again partial feedback. The advantages of partial feedback over nonfeedback or full feedback are more dramatic for the multifacility problem than they are for the single-facility. The partial feedback operator used is $\overset{E}{\Pi}$ where

$$(\overset{E}{\Pi}V)(i,J) = \min \left\{ \begin{array}{l} \min\limits_{k=1,\ldots,i} \left\{ \alpha_0 f_{ik} + w_{jk} + \beta \sum\limits_{\ell=1}^{N} p(J,L) \cdot (\overset{E}{\Pi}V)(k,L) \right\}, \\ \min\limits_{k=i+1,\ldots,N} \left\{ \alpha_0 f_{ik} + w_{jk} + \beta \sum\limits_{\ell=1}^{N} p(J,L) \cdot v(k,L) \right\} \end{array} \right. \quad (1.18)$$

A FORTRAN subroutine for policy improvement using $\overset{E}{\Pi}$ is listed in the Appendix. There is a scratch array of N elements which stores the terms, representing service cost and discounted future cost, that are independent of i in (1.18). A nonfeedback routine would require a scratch array of N^{M+1} elements with which to distinguish V from ΠV. The time per iteration would be about the same for nonfeedback and partial feedback; whereas, full feedback requires separate computation of the discounted future term for each i,J,k combination. Successive overrelaxation is used for value determination. A code for this procedure when $M = 3$ appears in the Appendix.

1.4 Computational Experience

Computational experience with the exact procedure for the multi-facility problem is reported in Table 3. In all cases the number of existing facilites or customers is $M = 3$ and the discount factor is $\beta = .9$. Values of N, the number of possible locations, ranged from 2 to 6; consequently the number of states ranged from N^{M+1}=16 to 1296.

Optimal policies were determined for each problem generated. The method of data generation is discussed in the Appendix. Through wide variation of α_m within and among problems, the optimal policies were caused to cover the entire "viscosity" range of server behavior from staying completely fixed to tight pursuit of the most critical customer.

The most commonly encountered optimal solutions involved loose pursuit of the α_m-weighted location centroid, i.e., relocations to follow vaguely the center of the present and near-future locations of those customers with highest interaction costs. In these and all other experiments, α_m variation had no discernible effects on computation load. The results reported in the table for each problem size are: the average and maximum number of policy improvements, the distribution of the number of value determinations and the average computation time (CPU seconds on a Univac 1108 computer) expended in each of these operations.

Table 3. Multifacility Computational Experience

		POLICY IMPROVEMENT			VALUE DETERMINATION			
		Number of PIR's			No. of problems for which no. of VDO's was:			
N	Number of Problems	Avg.	Max.	CPU time per PIR (sec)	0	1	2	Average CPU time per VDO (sec)
2 (16 states)	20	2.60	4	.009	0	19	1	.106
3 (81 states)	14	3.50	5	.089	0	11	3	1.002
4 (256 states)	13	4.69	8	.538	1	12	0	13.784
5 (625 states)	6	4.67	6	1.147	3	3	0	18.652
6 (1296 states)	6	5.50	7	6.816	0	6	0	245.020

Comparison of the computation times per iteration for PIR's and VDO's demonstrates why the algorithm selected is one that makes scant use of VDO's. Another important result is that, as was our experience with the single facility problem, the number of PIR's did not increase with problem size.

In all but 4 of the 59 problems solved, the number of VDO's was 0 or 1, indicating that in 55 problems the policy given by the first policy convergence was optimal. That is, the optimal policy was found before the first value determination, and the time required to verify optimality was far greater than the time to find the optimal solution. For instance, the longest-running of the 1296-state problems had policy convergence after 6 policy improvements of about 7 seconds each, and then a 4-minute value determination and one more policy improvement were required to verify that the solution was optimal.

2. A Surrogate Single Facility Approximation

For the remainder of this chapter we shall eschew the exact, multidimensional state-change approach to the multifacility location problem. This capitulation to the "curse of dimensionality" is made because of the difficulty not only in computing the optimal policy but also in *storing* it. To optimally control the process one needs to have available the prescribed decision for every state $(i,J)\epsilon N^{M+1}$, an array of N^{M+1} decisions. Retrieval on a long-term, once-per-period basis of the datum required in each time period may pose a serious information control problem. As another practical matter, with such an immense state space the percentage of states that are visited in any duration of interest is

extremely small. Thus, the bulk of decisions $k(i,J)$, even if they could be efficiently stored and retreived, would never actually be needed. Consequently, we shall propose approximation methods in the next two sections.

The approach suggested in this section is to replace the M-tuple of existing facility locations with a *surrogate* single-facility location. The problem is then solved as a single-facility problem employing the techniques of Chapter III, where the surrogate facility in some sense represents or replaces all the existing facilities. An overview of this approach is:

1. Define a mapping $\sigma: N^M \to N$ where $\sigma(J)$ is the surrogate for J.
2. Develop an approximate Markov chain model of the stochastic process σ_t, $t=0,1,2,\ldots$; where

$$\sigma_t \equiv \sigma(A_{1t},\ldots,A_{Mt}) \quad . \tag{2.1}$$

3. Solve the single-facility location problem

$$\text{Minimize } E\left\{\sum_{t=1}^{\infty}\left(\alpha_0 f(X_{t-1},X_t) + f'(X_t,\sigma_t)\right)\beta^{t-1}\right\} \tag{2.2}$$

where F' is a cost matrix for serving the surrogate facility.

4. Operate the multifacility system according to the following policy: Move the new facility to location $k(i,\sigma(J))\varepsilon N$ whenever state (i,J) is observed, where $\{k(\cdot,\cdot)\}$ is the optimal pure stationary policy obtained in Step 3.

There are three questions that need to be considered before this approach can be implemented: 1) How to choose σ ? 2) What to assume for the transition probabilities $\Pr\{\sigma_{t+1}=k|\sigma_t=i\}$? and 3) What is F' ? We shall address these issues in order in the remainder of this section.

In the presence of perfect information about the locations of existing facilities and in the absence of relocation cost, the location problem reduces to placing the new facility in each period at the point from which it can optimally serve the existing facilities. Methods for determining these points have been developed for a variety of situations [15]. In the present situation, with respect to existing facilities at locations $J = (j_1,\ldots,j_M)$, the new facility location that minimizes service cost is the minimizing index in the expression

$$\underset{k=1,\ldots,N}{\text{minimum}} \; W(J,k) \quad .$$

Let $\sigma(J)\epsilon N$ such that

$$W(J,\sigma(J)) \leq W(J,k) \qquad , k\epsilon N \quad .$$

Intuitively, $\sigma(J)$ is thought of as the "central location" or "centroid" when the existing facilities have location J . Of course, $\sigma(J)$ does not convey as much information as J , just as a sample mean conveys less information than the data it represents; but it does correspond to the point from which the new facility can expect to serve the existing facilities at minimal cost over one time period.

The surrogate single facility approach, i.e. replacing the set of existing facilities by their centroid and applying the techniques of Chapter III, is approximate because the changes over time of the centroid location $\sigma(J)$ are not necessarily Markovian. The discrete stochastic process σ_t , t=0,1,2,..., is an example of a *lumped process* [28]. The lumped process is induced from the multi-dimensional Markov chain $(A_{1t},\ldots,A_{Mt})$, t=0,1,2,..., through a partitioning, $C_1,\ldots,C_N$, of the state space of the chain, where

$$C_k = \{J\epsilon N^M | \sigma(J)=k\} \quad , k=1,\ldots,N \ .$$

That is, all N-tuple locations having the same centroid are "lumped" together to define a process with a smaller number of states than the original chain. For $J\epsilon N^M$, $k\epsilon N$ let

$$q(J,k) = \sum_{L\epsilon C_k} p(J,L) \ ; \tag{2.3}$$

if C_k is empty let $q(J,k)=0$. Note

$$q(J,k) = Pr\{\sigma_{t+1}=k | (A_{1t},\ldots,A_{Mt})=J\}$$

for all t=0,1,2,.... According to a theorem of Kemeny and Snell [28], the lumped process σ_t , t=0,1,2,... is a Markov chain if and only if for every pair of locations $i,k\epsilon N$, the values $q(J,k)$ are the same for every $J\epsilon C_i$. This *lumpability* condition, in other words, is that the centroid at time t+1 depends on the centroid, not on the particular existing facility locations, at time t . If this condition is satisfied,

we let $p'_{ik} = q(J,k)$ where J is any member of C_i ($p'_{ik} = 0$ if $C_i = \phi$), and we then have centroid transition probabilities

$$p'_{ik} = Pr\{\sigma_{t+1}=k \mid \sigma_t=i\} \quad , \; i,k\epsilon N, \; t=0,1,\ldots . \tag{2.4}$$

When the lumpability condition does not hold, an approximate Markov analysis of the lumped process may be performed using simulation to estimate centroid transition probabilities.

A possible choice for F', the service cost function for the surrogate facility, is a weighted sum of service costs for all the N-tuple locations associated with a particular centroid location, i.e. given new facility location k and centroid location j,

$$f'(k,j) = \sum_{J\epsilon C_j} \gamma(J)W(J,k) \quad , \; j,k\epsilon N \; . \tag{2.5}$$

The weights $\gamma(J)$ may be simply the reciprocal of the cardinality of C_j, resulting in an arithmetic average, or, at the expense of additional calculation, weights based on stationary probabilities.

3. Structured Policies

The second approximate approach proposed for the multifacility problem is a structured policy. As in the previous section we call upon our ability to solve single-facility problems; in this section, however, we solve M single-facility problems rather than one.

The structured policy is a decision rule that considers each customer individually to determine the optimal server location if the customer under consideration were the only customer; then it selects a

"compromise" server location based on the separate optima.

The single-facility problems which are solved at the outset of the structured policy procedure for $m=1,\ldots,M$ have extremal equations

$$v_m(i,j) = \min_{k\in N}\left\{\frac{\alpha_0}{M\alpha_m}f(i,k) + w_m(j,k) + \beta\sum_{\ell\in N} P_m(j,\ell)v_m(k,\ell)\right\} , \quad (3.1)$$

$(i,j)\in N\times N$. The optimal policy or *subpolicy* for the m^{th} problem, is designated $K_m=\{k_m(i,j)\}$. Problem (3.1) is identical to problem (2.17) of Chapter III except that the transition matrix P is replaced by P_m and the relocation cost factor α is replaced by $\alpha_0/M\alpha_m$. The ratio α_0/α_m is the ratio of relocation cost to service cost with respect to the m^{th} facility. The relocation cost factor for the single facility subproblem is scaled down to $\alpha_0/M\alpha_m$; the effect of this transformation is to give service costs in the subproblem the same relation to relocation costs that exists in the original M-customer problem.

Once the subpolicies K_m are computed and tabulated, the structured policy K_τ is defined in terms of a mapping $\tau:N^M\to N$ where

$$k_\tau(i,J) = \tau\left(k_1(i,j_1),\ldots,k_M(i,j_M)\right) \quad (3.2)$$

is the location at which the server should be located whenever state (i,J) is observed. The mapping τ is called the *compromise function* because for any given state (i,J) it transforms the set of recommended server locations, $k_1(i,j_1),\ldots,k_M(i,j_M)$, into a single compromise location $k_\tau(i,J)\in N$. The compromise function that has given good results on example problems small enough to be solved exactly is: $k_\tau(i,J) =$

the minimizing index in the expression

$$\min_{h \in N} \sum_{m=1}^{M} \alpha_m f\left(h, k_m(i, j_m)\right) \quad . \tag{3.3}$$

For six three-customer problems the structured policy was computed, and its associated values were determined by (1.13) using successive over-relaxation. These state values were compared with the optimal solution values obtained by the exact procedure of Section 1.3. For each problem Table 4 reports the percentage difference between the optimal solution values, V^* , and the structured policy values, V_τ . Average percent difference is 100 times

$$\frac{1}{N^M} \sum_{(i,J) \in N^{M+1}} \left| \frac{V_\tau(i,J) - V^*(i,J)}{V^*(i,J)} \right| \quad ;$$

maximal percentage difference is 100 times

$$\max_{(i,J) \in N^{M+1}} \left| \frac{V_\tau(i,J) - V^*(i,J)}{V^*(i,J)} \right| \quad .$$

Table 4. Computational Experience with a Structured Policy in the Multifacility Problem

Problem Number	N	$\alpha_1,\alpha_2,\alpha_3$	β	Percentage Difference in State Values Between Optimal and Structured Policies Average	Maximum
1	3	2, 4, 6	.9	1%	10%
2	3	1, 1, 1	.9	5%	23%
3	3	.2,.5,.7	.9	1%	10%
4	6	2, 3, 5	.9	1%	6%
5	6	1, 1, 1	.9	6%	11%
6	6	.8,1.2,1.4	.9	1%	4%

The difference in computational effort between the exact procedure and the structured policy is several orders of magnitude. With N=6 the solution of three single-facility problems is accomplished in a matter of milliseconds (as demonstrated by Table 2) whereas the exact procedure for a three-facility problem requires something closer to 5 minutes. The structured-policy approach is also very frugal in storage. Assuming that access to the subpolicies K_m, m=1,...,M and to the routines for computing $k_\tau(i,J)$ by (3.2) and (3.3) remains available during the life of the decision problem, it is not necessary to compute and store the entire structured policy array; rather, the decisions $k_\tau(i,J)$ can be obtained only as needed.

The heuristic, structured policy presented in this section is computationally tractable for virtually any size problem of interest. Because the multifacility problem is so difficult to solve exactly, it is impossible to experiment extensively on how near to optimal the

structured policy is. The limited experience we report is promising. It may be conjectured, from the similarity of a truly large multi-facility problem to problems in statistical mechanics, that the accuracy of the structured policy may increase with problem size; the computational experience reported is not inconsistant with this conjecture.

CHAPTER V

CONCLUSIONS AND EXTENSIONS

The research reported in this dissertation introduces the concepts and methodologies of stochastic decision processes into the modeling and solution of location analysis problems. The main conclusion is that it is beneficial to view certain common problems in location analysis as stochastic decision processes and that the resulting formulations of the location problems can be solved. Multidimensional Markov decision models of dynamic probabilistic location problems have been developed and solved. The modeling and solution procedures are straightforward, and the solutions exhibit a key behavior of service facilities--that of partly following, partly anticipating the moves of customers.

Computational loads are high. The key developments that allow solution are 1) the delayed-evaluation macro procedure, 2) the efficiently obtained hypermyopic starting values (for the single-facility problem), 3) the concept of partial feedback for policy improvement, 4) the exploitation of structure for value determination, and 5) the promising structured policy for large multifacility problems. Another important development is 6) the natural decomposition algorithm for rapid identification of the ergodic structure in an arbitrary Markov chain.

It is hoped that, given this work, more complex stochastic decision processes can be applied to location analysis. The following proposals for future work are suggested.

Distinct Location Spaces for Each Facility. In all the problems we

have considered it was assumed that the locations of all facilities, while changing over time, were confined to a set N. There is no conceptual difficulty in dealing with the situation where each facility has its own set of possible locations; say,

$$X_t \in N_0 = \{h_1, h_2, \ldots, h_{N_0}\}$$

$$A_{mt} \in N_m = \{g_{m_1}, g_{m_2}, \ldots, g_{m_{N_m}}\} \quad , \; m=1,\ldots,M \; .$$

A typical state $(i, j_1, \ldots, j_M) \in \mathop{\mathbf{X}}_{m=0}^{M} N_m$ refers to the condition of having the new facility at location h_i, and existing facility m at location $g_{m_{j_m}}$, $m=1,\ldots,M$. The transition probability matrices P_m are of dimension N_m; the relocation cost matrix F is $N_0 \times N_0$ and the service cost matrices F_m' are $N_0 \times N_m$. All the analysis of Chapter IV can be easily modified; for instance equation (1.7) becomes

$$W_m = P_m F_m'^{T} \; .$$

Efficient data management for the problem in this form is the key issue in developing this extension.

Multiple Servers. Consider dynamic, probabilistic problems in location analysis where there are several new facilities whose movements over time can be determined by choice. Let X_{st} be the location of the s^{th} server at time t, let γ_s be the relocation cost factor for the s^{th} server, and assume that customers are always served by the closest server; otherwise let the problem be identical to the one introduced in

Chapter IV. The problem is to devise a procedure for choosing $(X_{1,t}, \ldots, X_{St})$ given $(X_{1,t-1}, \ldots, X_{S,t-1}, A_{1,t-1}, \ldots, A_{M,t-1})$ so as to minimize the expected value of

$$\sum_{t=1}^{\infty} \left\{ \sum_{s=1}^{S} \gamma_s f(X_{s,t-1}, X_{st}) + \sum_{m=1}^{M} \alpha_m \min_{s=1,\ldots,S} f(X_{st}, A_{mt}) \right\} \beta^{t-1} .$$

Pursuit Situations. The single-facility model of Chapter III is similar to pursuit situations except for the fact that the pursued entity (existing facility) makes moves independent of the location of the pursuer (new facility). One way to enrich this model is to allow the location A_t to be dependent on X_{t-1} as well as on A_{t-1}. One can hope for reasonable results because the framework would still be Markovian, and no additional states would be generated. The complete transition matrix T would, however, lose its special structure.

Traveling Salesman Service. Let the cost of service in the single-server, multiple customer problem depend on the order in which customers are served. That is, let the peripatetic server take traveling-salesman tours rather than Weber tours.

Bayesian Estimation of Transition Matrices. Let the original transition matrices be updated through Bayesian learning as the process evolves.

Introduction of Queueing Phenomena. In all of the problems considered, the server has served all customers each period. An enrichment would be to relax this assumption and to give explicit consideration to the location-dependent time needed for service, i.e., to introduce queueing considerations into the model.

APPENDIX A

PROCEDURES FOR GENERATING RANDOM PROBLEMS

The following subroutine was used to generate random single-facility location problems. Multifacility problems were generated by a similar program which generates M stochastic matrices instead of one.

The distance matrix DIST (corresponding to F in the text) was generated by computing the rectilinear distance between pairs of points whose coordinates were uniformly distributed on (0,10) . The distances were truncated to the nearest hundredth of a unit in order to facilitate hand verification of computer solutions.

As noted in Chapter II Section 6.3 one of the procedures used by MacQueen and Hitchcock [33] for generating stochastic matrices leads to nearly uniform entries. The procedure presented here overcomes that difficulty. In MacQueen and Hitchcock's procedure "the row entries were randomly generated by selecting for each entry a uniformly distributed random number between 0 and 1 and then normalizing by the row total." [33] We begin as MacQueen and Hitchcock do by generating a row of uniform random numbers; however, before normalizing, each number is raised to a user-provided exponent. The effect of the exponent is to accentuate the differences among the uniform variates. For instance suppose the random numbers used to generate the first row of a 5-state transition matrix are .1554, .1372, .6132, .7417, .7503. The distributions resulting from various values of the exponent are tabled below:

EXPNT	1	2	3	4	5
1	.0648	.0573	.2557	.3093	.3129
2	.0158	.0123	.2454	.3591	.3674
5	.00017	.00009	.1579	.4088	.43304
10	.0	.0	.0656	.4401	.4943
500	.0	.0	.0	.0031	.9969

By use of this procedure test matrices over a wide structural range can be generated. In the FORTRAN code below probabilities that would be very small are set to zero.

```
      SUBROUTINE GENDAT(NMAX,NODES,EXPNT,KSEED,PROB,DIST)
      DIMENSION PROB(NMAX,NMAX),DIST(NMAX,NMAX)
      N1=NODES-1
      DO 10 I=1,NODES
      PROB(I,1)=10.0*RANDOM(KSEED)
10    PROB(I,2)=10.0*RANDOM(KSEED)
      DO 20 I=1,N1
      J1=I+1
      DO 30 J=J1,NODES
      DIST(I,J)= ABS(PROB(I,1)-PROB(J,1)) +
     + ABS(PROB(I,2)-PROB(J,2))
      DIST(I,J)=.001*INT(1000.0*DIST(I,J) +.5)
30    DIST(J,I)=DIST(I,J)
20    DIST(I,I)=0.
      DIST(NODES,NODES)=0.
C
C
      UNDER=1.0E-28
      CUT=UNDER**(1./EXPNT)
      DO 100 I=1,NODES
      S=0
      DO 200 J=1,NODES
      R=RANDOM(KSEED)
      IF(R .GT. CUT) GO TO 300
      PROB(I,J)=0.
```

```
      GO TO 200
300   PROB(I,J)=R**EXPNT
      S=S+PROB(I,J)
200   CONTINUE
      PTEST=0.
      DO 400 J=1,NODES
      PROB(I,J)=.001*INT(1000.*PROB(I,J)/S +.5)
400   PTEST=PTEST+PROB(I,J)
      DIF=1.0-PTEST
      IF (ABS(DIF) .LT. .00001) GO TO 100
      DO 500 J=1,NODES
      PP=PROB(I,J)+DIF
      IF(PP .GT. 1. .OR. PP .LT. 0.) GO TO 500
      PROB(I,J)=PP
      GO TO 100
500   CONTINUE
100   CONTINUE
      RETURN
      END
```

APPENDIX B

FORTRAN CODE FOR NATURAL DECOMPOSITION

The following subprogram solves the system of equations

$$v_i = r_i + \beta \sum_{j=1}^{N} p_{ij} v_j \quad , \quad i=1,\ldots,N$$

for a nonergodic valued Markov chain by the state classification (natural decomposition) algorithm of Section 3.2.3 of Chapter III. The program input is N,β,R and P; the output is V; other variables are used internally.

```
      SUBROUTINE FVDOSUB(N,BETA,R,P,V,
     +B,IC,ISET,NLEAD,ILEAD)
      REAL P(N,N),V(N),R(N)
      LOGICAL B(N,N)
      INTEGER IC(N),ISET(N),NLEAD(N),ILEAD(1000)
      IDENSE=0
      ICOUNT=0
      DO 10 I=1,N
      DO 20 J=1,N
      B(I,J)=(P(I,J) .GT. 0.0)
      IF (.NOT. B(I,J)) GO TO 20
      IDENSE=IDENSE+1
20    CONTINUE
      B(I,I)=.FALSE.
10    ISET(I)=I
C     CHECK FOR ABSORBING STATES
      DO 40 I=1,N
      IF (P(I,I) .LT. 1.0) GO TO 40
      V(I)=R(I)/(1-BETA)
      B(I,I)=.TRUE.
      ICOUNT=ICOUNT+1
40    CONTINUE
      IF (ICOUNT .EQ. N) GO TO 1000
C  **ALL STATEMENTS ABOVE THIS LINE ARE EXECUTED ONLY ONCE**
30    CALL ZEROW(N,B,IZEROW)
      IF (IZEROW .GT. 0) GO TO 50
      CALL CYCLE(N,B,ISET,IZEROW,IC)
50    CALL SOLVE(N,IZEROW,P,V,R,BETA,ICOUNT,IC,ISET,B,NLEAD,
     +IDENSE)                                          ILEAD,
      IF (ICOUNT .LT. N) GO TO 30
1000  RETURN
```

```
      SUBROUTINE CYCLE (N,B,   SET,IZEROW,IC)
      LOGICAL B(N,N)
      INTEGER SET(N),IC(N)
C     B(I,I) =.I IS DELETED.
C     SET(I)=0 IFF I IS IN THE PATH
      NC=0
C     FIND FIRST STATE
      DO 10 J=1,N
      IF (B(J,J)) GO TO 10
      IL=J
      GO TO 30
10    CONTINUE
C     ADD STATE TO PATH
30    NC=NC+1
      IC(NC)=IL
      SET(IL)=0
C     FIND ANOTHER STATE% IF NOT A REPEAT ADD TO PATH
      DO 40 J=1,N
      IF (B(J,J) .OR. .NOT. B(IL,J)) GO TO 40
      IL=J
      GO TO 50
40    CONTINUE
50    IF (SET(IL).NE. 0) GO TO 30
C     REPEAT FOUND. FIND FIRST OCURRENCE
      IR=IL
      DO 70 I=1,NC
      IF (IC(I) .NE. IR) GO TO 70
      NR=I
      GO TO 80
70    CONTINUE
C     CYCLE IDENTIFIED  IR=IC(NR),IC(NR+1),...,IC(NC),IR
C     COLLAPSE CYCLE
80    B(IR,IR)=.TRUE.
      NR1=NR+1
      DO 100 I=NR1,NC
      JR=IC(I)
      B(JR,JR)=.TRUE.
      SET(JR)=IR
      DO 110 K=1,N
      IF (SET(K) .EQ. JR) SET(K)=IR
      IF (B(K,K)) GO TO 110
      B(IR,K)=B(IR,K) .OR. B(JR,K)
      B(K,IR)=B(K,IR) .OR. B(K,JR)
110   CONTINUE
100   CONTINUE
C     CHECK ROW(IR) FOR ZEROW
      DO 120 J=1,N
```

```
      IF (B(J,J) .OR. .NOT. B(IR,J)) GO TO 120
      IL=J
      B(IR,IR)=.FALSE.
      NC=NR
      GO TO 50
120   CONTINUE
      DO 140 I=1,NR
      JR=IC(I)
140   SET(JR)=JR
      IZEROW=IR
      RETURN
      END



      SUBROUTINE SOLVE(N,IZEROW,P,V,R,BETA,ICOUNT,IC,SET,B
     +ILEAD,IDENSE)                                        ,NLEAD,
      LOGICAL B(N,N),DIF
      INTEGER IC(N),SET(N),NLEAD(N),ILEAD(IDENSE)
      REAL P(N,N),V(N),R(N)
      IZ=IZEROW
C     DETERMINE SIZE OF SYSTEM & IDENTIFY VARIABLES
      ISIZE=0
      DO 10 I=1,N
      IF (SET(I) .NE. IZ) GO TO 10
      ISIZE=ISIZE+1
      B(I,I)=.FALSE.
      IC(ISIZE)=I
10    CONTINUE
      IF (ISIZE .GT. 1) GO TO 30
      SUM=0.0
      DO 20 I=1,N
20    IF (B(IZ,I)) SUM=SUM+P(IZ,I)*V(I)
      V(IZ)=(R(IZ)+BETA*SUM)/(1-BETA*P(IZ,IZ))
      B(IZ,IZ)=.TRUE.
      ITER=0
      GO TO 190
C     UPDATE RIGHT-HAND-SIDE
30    DO 50 I=1,ISIZE
      JR=IC(I)
      SUM=0.0
      DO 40 K=1,N
      IF (.NOT. B(JR,K)) GO TO 40
      IF (SET(K) .EQ. IZ) GO TO 40
      SUM=SUM+P(JR,K)*V(K)
40    CONTINUE
      R(JR)=R(JR)+BETA*SUM
50    CONTINUE
C     IDENTIFY NONZERO TERMS IN SYSTEM
```

```
      JNL=0
      DO 80 I=1,ISIZE
      IR=IC(I)
      SUM=0.0
      NL=0
      DO 110 J=1,ISIZE
      JR=IC(J)
      IF (P(IR,JR) .LE. 0.0) GO TO 110
      NL=NL+1
      JNL=JNL+1
      ILEAD(JNL)=JR
110   CONTINUE
      NLEAD(I)=NL
      B(IR,IR)=.TRUE.
80    CONTINUE
      IF (JNL .GT. IDENSE) GO TO 220
      MXIT=1000
C     SOR
      ITER=1
120   DIF=.FALSE.
      ITER=ITER+1
      JNL=0
      DO 130 I=1,ISIZE
      IR=IC(I)
      SUM=0.0
      NL=NLEAD(I)
      DO 140 J=1,NL
      JNL=JNL+1
      JR=ILEAD(JNL)
140   SUM=SUM+P(IR,JR)*V(JR)
      SUM=1.3*(R(IR)+BETA*SUM)-.3*V(IR)
      DIF=DIF .OR. (ABS(SUM-V(IR)) .GT. .00001)
      V(IR)=SUM
130   CONTINUE
      IF (.NOT. DIF) GO TO 190
      IF (ITER .GT. MXIT) GO TO 180
      GO TO 120
180   WRITE(6,181) MXIT
181   FORMAT(18H SOR STOPPED AFTER,I8,2X,11H ITERATIONS)
190   ICOUNT=ICOUNT+ISIZE
      DO 200 I=1,ISIZE
      IR=IC(I)
200   WRITE(6,201) IR,V(IR)
201   FORMAT (I8,F12.6)
      WRITE(6,202) ITER
202   FORMAT (I8,2X,15H SOR ITERATIONS)
      RETURN
220   WRITE(6,221) JNL
221   FORMAT (31H DIMENSION OF ILEAD MUST EXCEED,I10)
      STOP
      END
```

```
      SUBROUTINE ZEROW(N,B,IZEROW)
      LOGICAL B(N,N)
C     IZEROW IS A NONDELETED STATE THAT DOES NOT
C          LEAD TO ANY OTHER NONDELETED STATE, IF IT EXISTS
C          OTHERWISE IZEROW=0.
          DO 120 I=1,N
120       WRITE(6,105)(B(I,J),J=1,N)
105       FORMAT(10L7)
      DO 10 I=1,N
      IF (B(I,I)) GO TO 10
      DO 20 J=1,N
      IF (B(J,J) .OR. .NOT. B(I,J)) GO TO 20
      GO TO 10
20    CONTINUE
      IZEROW=I
      RETURN
10    CONTINUE
      IZEROW=0
      RETURN
      END
```

The system of value equations for a particular canonical class is solved in the sequence of code following the comment SOR and up to line 180 in subroutine SOLVE. The method of solution is a successive overrelaxation routine using indexing to eliminate all multiplications by zero. Any other method for solving the system can be substituted in place of this routine. If it is desired to let the size of the canonical class determine the method to be used then the relevant branching variable is ISIZE.

APPENDIX C

PARTIAL FEEDBACK POLICY IMPROVEMENT ROUTINE FOR THE MULTIFACILITY LOCATION PROBLEM

```
      SUBROUTINE PIR4(NMAX,NODES,PROB,DIST,VALUE,KPOLCY,
     +                VISIT,BETA,WORK,KCNVRG)
      DIMENSION PROB(3,NMAX,NMAX),DIST(NMAX,NMAX),
     +          VALUE(NMAX,NMAX,NMAX,NMAX),
     +          KPOLCY(NMAX,NMAX,NMAX,NMAX),
     +          VISIT(NMAX,NMAX,NMAX,NMAX)
     +          WORK(NMAX,NMAX,NMAX,NMAX)
C *** PARTIAL FEEDBACK POLICY IMPROVEMENT ***
C *** WORK = EXPECTED (SERVICE + DISCOUNTED FUTURE) COST
      KCNVRG=1
      DO 50 J3=1,NODES
      DO 50 J2=1,NODES
      DO 50 J1=1,NODES
      DO 40 K=1,NODES
      S=0.0
      DO 30 L3=1,NODES
      IF (PROB(3,J3,L3) .LE. 0.0) GO TO 30
      DO 20 L2=1,NODES
      IF (PROB(2,J2,L2) .LE. 0.0) GO TO 20
      DO 10 L1=1,NODES
      IF (PROB(1,J1,L1) .LE. 0.0) GO TO 10
      S=S+PROB(1,J1,L1)*PROB(2,J2,L2)*PROB(3,J3,L3)*
     +VALUE(K,L1,L2,L3)
10    CONTINUE
20    CONTINUE
30    CONTINUE
40    WORK(J1,J2,J3,K)=VISIT(J1,J2,J3,K)+BETA*S
      DO 50 I=1,NODES
      VNEW=DIST(I,1)+WORK(J1,J2,J3,1)
      KNEW=1
      DO 60 K=2,NODES
      V1=DIST(I,K)+WORK(J1,J2,J3,K)
      IF(VNEW .LE. V1) GO TO 60
      VNEW=V1
      KNEW=K
60    CONTINUE
      IF (KNEW .EQ. KPOLCY(I,J1,J2,J3)) GO TO 70
      KPOLCY(I,J1,J2,J3)=KNEW
      KCNVRG=0
70    VALUE(I,J1,J2,J3)=VNEW
50    CONTINUE
      RETURN
```

APPENDIX D

SUCCESSIVE OVERRELAXATION VALUE DETERMINATION OPERATION

FOR THE MULTIFACILITY LOCATION PROBLEM

```
      SUBROUTINE SORVD4(NMAX,NODES,PROB,DIST,VALUE,KPOLCY,
     +                  VISIT,BETA,EPSLON)
      DIMENSION PROB(3,NMAX,NMAX),DIST(NMAX,NMAX),
     +          VALUE(NMAX,NMAX,NMAX,NMAX),
     +          KPOLCY(NMAX,NMAX,NMAX,NMAX),
     +          VISIT(NMAX,NMAX,NMAX,NMAX)
      ROMEGA=1.6
      SOMEGA=1.-ROMEGA
      Z=3HNOT
      ITER=0
50    ITER=ITER+1
      IF (ITER .GE. 50) GO TO 70
      ERROR=0.0
      DO 40 I=1,NODES
      DO 40 J3=1,NODES
      DO 40 J2=1,NODES
      DO 40 J1=1,NODES
      K=KPOLCY(I,J1,J2,J3)
      S=0.0
      DO 30 L3=1,NODES
      IF (PROB(3,J3,L3) .LE. 0.0) GO TO 30
      DO 20 L2=1,NODES
      IF (PROB(2,J2,L2) .LE. 0.0) GO TO 20
      DO 10 L1=1,NODES
      IF (PROB(1,J1,L1) .LE. 0.0) GO TO 10
      P = PROB(1,J1,L1)*PROB(2,J2,L2)*PROB(3,J3,L3)
      S=S+P*VALUE(K,L1,L2,L3)
10    CONTINUE
20    CONTINUE
30    CONTINUE
      V1=DIST(I,K)+VISIT(J1,J2,J3,K)
      V1=(V1+BETA*S)*ROMEGA + SOMEGA*VALUE(I,J1,J2,J3)
      V2=ABS((V1-VALUE(I,J1,J2,J3))/V1)
      IF (V2 .GT. ERROR) ERROR=V2
      VALUE(I,J1,J2,J3)=V1
40    CONTINUE
      IF (ERROR .GT. EPSLON) GO TO 50
      Z=3H
70    WRITE(6,1) Z,ITER,ERROR
1     FORMAT(" SOR DID",A3," CONVERGE",I10," ITERATIONS",
     +       E10.2,"  MAXIMUM ERROR")
      RETURN
```

BIBLIOGRAPHY

1. Aly, A. A., "Probabilistic Formulations of Some Facility Location Problems," PhD dissertation, Virginia Polytechnic Institute and State University (1975)

2. Ballou, R., "Dynamic Warehouse Location Analysis," Journal of Marketing Research, Vol. 5, 271-6 (1968)

3. Bell, C. E., "Characterization and Computation of Optimal Policies for Operating an M/G/1 Queueing System with Removable Server," Operations Research, Vol. 19, 208-18 (1971)

4. Bell, C. E., "Turning Off a Server with Customers Present: Is This Any Way to Run an M/M/c Queue with Removable Servers?" Operations Research, Vol. 23, 571-4 (1975)

5. Chapman, S. C. and J. A. White, "Probabilistic Formulations of Emergency Service Facilities Location Problems," ORSA/TIMS National Meeting, San Juan PR (1974)

6. Çinlar, E., Introduction to Stochastic Processes, Prentice-Hall (1975)

7. Conte, S. D. and C. deBoor, Elementary Numerical Analysis, Second Edition, McGraw-Hill (1972)

8. Contreras, L. E., "Large-Scale Programming in Markov Decision Processes," PhD dissertation, School of Industrial and Systems Engineering, Georgia Institute of Technology (1975)

9. Cooper, L. "A Random Locational Equilibrium Problem," Journal of Regional Science, Vol. 14, 47-54 (1974)

10. deGhellinck, G. T. and G. D. Eppen, "Linear Programming Solutions for Separable Markovian Decision Problems," Management Science, Vol. 13, 371-94 (1967)

11. Denardo, E. V., "Contraction Mappings in the Theory Underlying Dynamic Programming," SIAM Review, Vol. 19, 165-77 (1967)

12. Derman, C. Finite State Markovian Decision Processes, Academic Press (1970)

13. Effroymson, M. A. and T. L. Ray, "A Branch-Bound Algorithm for Plant Location," Operations Research, Vol. 14, 361-68 (1966)

14. Fox, B. L. and D. M. Landi, "An Algorithm for Identifying the Ergodic Subchains and Transient States of a Stochastic Matrix," Communications of the ACM, Vol. 11, 619-21 (1968)

15. Francis, R. L. and J. A. White, Facility Layout and Location: An Analytical Approach, Prentice-Hall (1974)

16. Hadley, G. and T. M. Whitin, Analysis of Inventory Systems, Prentice-Hall (1963)

17. Hartman, J. K. and Lasdon, L. S., "A Generalized Upper Bounding Method for Doubly Coupled Linear Programs," Naval Research Logistics Quarterly, Vol. 17, 411-29 (1970)

18. Hastings, N. A. H., "Some Notes on Dynamic Programming and Replacement," Operational Research Quarterly, Vol. 19, 453-64 (1968)

19. Hastings, N. A. J., "Optimization of Discounted Markov Decision Problems," Operational Research Quarterly, Vol. 20, 499-500 (1969)

20. Hastings, N. A. J. and J. M. C. Mello, "Tests for Suboptimal Actions in Discounted Markov Programming," Management Science, Vol. 19, 1019-22 (1973)

21. Hillier, F. S. and G. J. Lieberman, Operations Research, Second Edition, Holden-Day (1974)

22. Howard, R. A., Dynamic Programming and Markov Processes, John Wiley (1960)

23. Howard, R. A., Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes, John Wiley (1971)

24. Hurter, A. P. and J. Prawda, "A Warehouse Location Problem with Probabilistic Demand," Working Paper No. 42, Graduate School of Business Administration, Tulane University

25. Jucker, J. V. and R. C. Carlson, "The Simple Plant Location Problem under Uncertainty," Technical Report No. 75-3, Department of Industrial Engineering, Stanford University (1975)

26. Katz, I. N. and L. Cooper, "An Always-Convergent Numerical Scheme for a Random Locational Equilibrium Problem," SIAM Journal of Numerical Analysis, Vol. 11, 683-91 (1974)

27. Katz, I. N. and L. Cooper, "Normally and Exponentially Distributed Locational Equilibrium Problems," Department of Applied Mathematics and Computer Science, Washington University (1974)

28. Kemeny, J. G. and J. L. Snell, Finite Markov Chains, D. Van Nostrand (1969)

29. Kushner, H. J. and C. H. Chen, "Decomposition of Systems Governed by Markov Chains," IEEE Transactions on Automatic Control, Vol. AC-19, 501-7 (1974)

30. Kushner, H. J. and A. J. Kleinman, "Accelerated Procedures for the Solution of Discrete Markov Control Problems," IEEE Transactions on Automatic Control, Vol. AC-16, 147-52 (1971)

31. MacQueen, J., "A Modified Dynamic Programming Method for Markovian Decision Problems," Journal of Mathematical Analysis and Applications, Vol. 14, 38-43 (1966)

32. MacQueen, J., "A Test for Suboptimal Actions in Markovian Decision Problems," Operations Research, Vol. 15, 559-61 (1967)

33. MacQueen, J. and D. F. Hitchcock, "On Computing the Expected Discounted Return in a Markov Chain," Naval Research Logistics Quarterly, Vol. 17 (1970)

34. Miller, B. L., "Dispatching from Depot Repair in a Recoverable Item Inventory System: On the Optimality of a Heuristic Rule," Management Science, Vol. 21 (1974)

35. Porteus, E. L., "Some Bounds on Infinite Horizon Discounted Sequential Decision Processes," Management Science, Vol. 18, 7-11 (1971)

36. Porteus, E. L., "On the Optimality of Structured Policies in Countable State Decision Processes," Research paper No. 141, Graduate School of Business, Stanford University (1974)

37. Porteus, E. L., "Bounds and Transformations for Discounted Finite Markov Decision Chains," Operations Research, Vol. 23, 761-83 (1975)

38. Prabhu, N. V. and S. Stidham, Jr., "Optimal Control of Queueing Systems," Mathematical Methods in Queueing Theory, ed. A. B. Clarke, Springer-Verlag (1974)

39. ReVelle, C., D. Marks and J. C. Liebman, "An Analysis of Private and Public Sector Location Models," Management Science, Vol. 16, 692-707 (1970)

40. Roodman, G. and L. B. Schwarz, "Optimal and Heuristic Facility Phase-out Strategies," AIIE Transactions, Vol. 7, 177-84 (1975)

41. Ross, S., Applied Probability Models with Optimization Applications, Holden-Day (1970)

42. Schweitzer, P. J., "Multiple Policy Improvements in Undiscounted Markov Renewal Programming," Operations Research, Vol. 19, 784-93 (1971)

43. Schweitzer, P. J., "Annotated Bibliography on Markov Decision Processes," IBM Research Center (1973)

44. Scott, A. J., "Dynamic Location-Allocation Systems: Some Basic Planning Strategies," Environment and Planning, Vol. 3, 73-82 (1971)

45. Serfozo, R. F., "Monotone Optimal Policies for Markov Decision Processes," Symposium on Stochastic Systems, University of Kentucky (1975)

46. Shapiro, J. F., "Brouwer's Fixed Point Theorem and Finite State Space Markovian Decision Theory," Journal of Mathematical Analysis and Applications, Vol. 49, 710-2 (1975)

47. Sheppard, E. S., "A Conceptual Framework for Dynamic Location-Allocation Analysis," Discussion Paper No. 14, Department of Geography, University of Toronto (1973)

48. Seppala, Y., "On a Stochastic Multifacility Location Problem," AIIE Transactions, Vol. 7, 56-62 (1975)

49. Swersey, A. J., "Models for Reducing Fire Engine Response Times," PhD dissertation, School of Engineering and Applied Science, Columbia University (1972)

50. Tapiero, C. S., "Transportation-Location-Allocation Problems over Time, Journal of Regional Science, Vol. 11, 377-86 (1971)

51. Totten, J. C., "Computational Methods for Finite State Finite Valued Markovian Decision Problems," Operations Research Center, University of California, Berkeley (1971)

52. Veinott, A. F., "Markov Decision Chains," Studies in Optimization, Vol. 10, MAA Studies in Mathematics, ed. G. B. Dantzig and B. C. Eaves (1974)

53. Viswanathal, B., V. V. Aggarwal and K. P. K. Nair, "Multiple Criteria Markov Decision Process," ORSA/TIMS Joint National Meeting, Chicago (1975)

54. Wesolowsky, G. O., "Dynamic Facility Location," Management Science, Vol. 20 (1973)

55. Wesolowsky, G. O. and W. G. Truscott, "The Multiperiod Location-Allocation Problem with Relocation of Facilities," McMaster University, Hamilton, Ontario (1972)

56. Westlake, J. R., A Handbook of Numerical Matrix Inversion and Solution of Linear Equations, John Wiley (1968)

57. White, D. J., "Dynamic Programming, Markov Chains, and the Method of Successive Approximations," Journal of Mathematical Analysis and Applications, Vol. 6, 373-6 (1963)

58. White, J. A., "Location Theory," lecture notes, School of Industrial and Systems Engineering, Georgia Institute of Technology (1975)

59. Wolfe, P. and G. B. Dantzig, "Linear Programming in a Markov Chain," Management Science, Vol. 10, 702-10 (1962)

60. Young, D., "Probabilistic Models in Operations Research," lecture notes, School of Industrial and Systems Engineering, Georgia Institute of Technology (1974)

61. Young, D., "Cost Modeling in Discrete Markov Chains," School of Industrial and Systems Engineering, Georgia Institute of Technology (1974)

62. Young, D. and L. E. Contreras, "Expected Present Worths of Cash Flows Under Uncertain Timing," The Engineering Economist, Vol. 20, 257-68 (1975)

63. Zaldivar, M. and T. J. Hodgson, "Rapid Convergence Techniques for Markov Decision Processes," Decision Sciences, Vol. 6, 14-24 (1975)

64. Symposium on Sparse Matrix Computations, Argonne National Laboratory (1975)

VITA

Richard E. Rosenthal was born in 1950, third son of Robert and Marjorie Rosenthal. His primary and secondary education was in the New York District #14 Public School System. He took an interest in things mathematical at an early age, gaining his first exposure to such topics as the calculus from tutoring by his older brothers.

Mr. Rosenthal entered the Johns Hopkins University with advanced standing in mathematics in 1968. His interest focused on operations research midway through college, and in 1972 he earned the B. A. (with honors) in quantitative sciences.

Following college Mr. Rosenthal entered the doctoral program in operations research at the School of Industrial and Systems Engineering, Georgia Institute of Technology. He completed the requirements for the Ph.D. in 1975 and accepted an appointment as Assistant Professor of Management Science in the College of Business Administration, University of Tennessee, Knoxville.

Mr. Rosenthal has part-time work and consulting experience in the fields of public transportation, law enforcement and production control.

In 1973 Mr. Rosenthal married Diana Elizabeth White who is a graduate of Emory University and a certified medical technologist.